Université Blaise Pascal
(Université Clermont Auvergne)
© 2016 A. Sigayret

Systèmes d'exploitation

(4) Scripts Microsoft

CM4 - 2016 v.1

1. Shell MSDOS (sous Windows)

Sous DOS

- Shell = interpréteur de commandes : command.com
  Scripts, variables (%PATH%, ...)
- Fichiers de script : BAT
  insensible à la casse : CALL, ECHO, FOR, GOTO, F, SET, ...
  variables avec % et %% (pas $)
  paramètres avec / (pas –)
  par défaut affichage des commandes
  (pour masquer, utiliser @ : @ECHO OFF)

Sous Windows

- Shell command com (jsq Win 9x) puis cmd.exe (Win NT)
  cmd.exe → Console Win32
- Fichiers de script : bat, .cmd
  + variables d'environnement : %WINDIR%, %TEMP%, ...
  ← héritage DOS
- + Fichiers de configuration : ini
  dont : boot.ini, desktop.ini, ...
  ← Win NT : + base de registre
  + autorun.inf

```
[autorun]
Icon=***
Open=***
Shell\item="***"
Shell\item\command=***
```


2. Langages de script pour Microsoft

Langages de script : automatisation simple et intégrée de tâches (tous S.E.)

VBS (Visual Basic Script)
- Sous-ensemble de Visual Basic
- Privilégié par Microsoft
  + bonne intégration à Windows
  + aussi pour l E., MS Office, ...
  → virus !!

JS
- Java Script + intégration à Windows

Perl
- originellement sous Unix

- **WSH** (Windows Scripting Host)
  → cscript exe (CLI) + wscript exe (GUI) + mshta exe (appli. HTML)
  → remplacé par PowerShell & .NET

+ Autres langages interprétés (tous S.E.)

3. PowerShell

- Langage de script
  - orienté objet (cf .NET)
  - jeu de commandes (*commandlets*) et de fonctions étendu
    - format verbe-nom (Exp. Move-Item = move-item)
    + alias commandes CMD/DOS (Exp. move)
    + alias commandes comme Unix (Exp. mv)
  - opérateurs et expressions (attention \ pour sous-dossiers)

- Interpréteur CLI interactif
  + programmation GUI

- Exécution de scripts en arrière-plan ou à distance,
  débogage, multilinguisme, ...

- version 1

Bibliographie :
- PowerShell sur Wikipedia & sur microsoft.com
- Windows PowerShell, A.Petitjean, R.Lemesle [ENI, 2006] → B U S

3. PowerShell

3.1. Aide et information

- Aide : help, get-help (options : -detailed, -full, MàJ : Update-Help)

- Information : get-command

Exp.

get-command –verb move

```
CommandType     Name                                ModuleName
-----------     ----                                ----------
Function        Copy-NetFirewallRule                NetSecurity
Function        Copy-NetIPsecMainModeCryptoSet      NetSecurity
Function        Copy-NetIPsecMainModeRule           NetSecurity
Function        Copy-NetIPsecPhase1AuthSet          NetSecurity
Function        Copy-NetIPsecPhase2AuthSet          NetSecurity
Function        Copy-NetIPsecQuickModeCryptoSet     NetSecurity
Function        Copy-NetIPsecRule                   NetSecurity
Cmdlet          Copy-Item                           Microsoft...
Cmdlet          Copy-ItemProperty                   Microsoft...
```

get-command –noun Error

```
CommandType     Name                                ModuleName
-----------     ----                                ----------
Cmdlet          write-Error                         Microsoft...
```

get-command rm

```
CommandType     Name                                ModuleName
-----------     ----                                ----------
Alias       rm -> Remove-Item
```


3. PowerShell

3.2. Objets et méthodes

Exp. objet [string] :

$machaine="bonjour chez vous!"

$machaine | get-member

```
   TypeName : System.String

   Name             MemberType          Definition
   ----             ----------          ----------
   Clone            Method              System.Object Clone(), System.Object ...
   CompareTo        Method              int CompareTo(System.Object value), i...
   Contains         Method              bool Contains(string value)
   CopyTo           Method              void CopyTo(int sourceIndex, char[] d...
   EndsWith         Method              bool EndsWith(string value), bool End...
   Equals           Method              bool Equals(System.Object obj), bool ...
   GetEnumerator    Method              System.CharEnumerator GetEnumerator()...
   GetHashCode      Method              int GetHashCode()
   GetType          Method              type GetType(
   ...
   ToUpper          Method              string ToUpper(), string ToUpper(cult...
   ...
   Length           Property            int Length {get;}
```

$machaine toupper()

```
   BONJOUR CHEZ VOUS!"
```

$machaine length

```
   18
```


3. PowerShell

3.3. Alias

- Alias comme Unix → Nom PowerShell ← Alias comme DOS
  → privilégier les commandes Unix ?

- commande : Get-Alias *nomalias*

- se méfier de la syntaxe :

  Move-Item –path .*.txt –destination ".\archives textes"

  move-item .*.txt .\"archives textes"

  mv .*.txt .\'archives textes'

  ...

- Création d'alias : New-Alias, Set-Alias, ...

3. PowerShell

3.3. Alias

Unix	*PowerShell*	*DOS*
cat	Get-Content	type
cd	Set-Location	chdir
clear	Clear-Host	
cp	Copy-Item	copy
echo	Write-Output	echo
kill	Stop-Process	
ls	Get-ChildItem	dir
	New-Item	md
mount	New-PSDrive	
mv	Move-Item	move
ps	Get-Process	
pwd	Get-Location	
mv	Rename-Item	ren
rm rmdir	Remove-Item	del rd
set	Set-Variable	
sleep	Start-Sleep	
sort	Sort-Object	
tee	Tee-Object	
	...	

3. PowerShell

3.4. Affichage

- Redirigeable : | > >>
  N B. : 2>&1 (cf stderr → stdout)

- Affichage par défaut d'un objet :
  - propriétés choisies par le système
  - sous forme de tableau

Exp. Get-ChildItem

```
Répertoire : C:\opt\cygwin6   home\Inv
Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d----          01/03/2016     10:01            bin
-----          18/12 2015     10:56        605  bashrc
-a---          09/03/2016     18.25       1018  .bash_history
-----          18/12 2015     10:56       1 9   bash_profi e
-----          18/12 2015     10:56       1919  inputrc
-----          18/12 2015     10:56       1236  prof le
...
```

- Possible :
  - autres formats :
    Format-List (fl),
    Format-Table (ft),
    Format-Wide (fw),
    Format-Custom (fc)
  - sélection de propriétés :
    –Property *p1 ... pk*
    toutes : *

Exp. Get-Child tem | fl –Property Name

```
Name   b n
Name   .bash  c
Name   .bash_h sto y
Name   .bash_p  of le
Name   . nput  c
Name   .p  of le
. .
```


3. PowerShell

3.5. Opérateurs, variables, expressions

"	délimiteur ($ métacaractère)
'	délimiteur ($ caractère)
$	initiateur de nom de variable
	caractère d'échappement (cf \ Unix) 0 (null), b (back), f (page), n (LF), r (CR), t (tab), v (vtab), ' " $
=	affectation de variable mais aussi : += –= *= /= %= ++ --
[]	tableau, transtypage
#	début de commentaire
()	délimiteurs d'expression
{}	délimiteurs de bloc

Exp. :

```
$ch="mot"
echo "`$ch"          → $ch
echo `$ch`           → $ch`
echo "$ch: longueur= $ch.Length"     → mot: longueur=mot.Length
echo "$ch: longueur= $($ch.Length)"  → mot: longueur=3

$t='a' ;  $v=[int][char]$t ;  $v     → 97
```


3. PowerShell

3.5. Opérateurs, variables, expressions

Constantes :
$True, $False

Variables :
$? : la dernière opération a-t-elle réussie ($True/$False)
$^ : premier mot de la dernière commande reçue par le shell
$$: dernier mot de la dernière commande reçue par le shell
$_ : objet courant transmis par un tube (|)
$Input : liste des objets transmis par un tube
$LastExitCode
$OFS : séparateur de champs pour conversion tableau-chaine

$Args : tableau des arguments passés à une fonction ou un script
$Error : tableau des erreurs

$Home, $Host, ...

chaine = tableau de caractères : "abcdef"[3] → 'c'
tableau : $t= 1, 2, 3, 'g', 4
$t[0] →1
$t Length → 4
$t=$t+9 → 1 2 3 g 4 9

3. PowerShell

3.5. Opérateurs, variables, expressions

Opérations :

- arithmétiques : + – * / %

- chaines, tableaux : +

- construction d'intervalle : **..**

- comparaison de valeur :
  - sans distinguer la casse : -eq -ne -gt -ge -lt -le
  - en distinguant la casse : -ceq -cne -cgt -cge -clt -cle

- comparaison de type : -is -isnot
  - usage : *donnée* –is *type* avec types [int], [char], [string]

- logiques : –not (aussi : !) –and –or –xor

- "binaires" (bit à bit) : –bnot –band –bor –bxor

3. PowerShell

3.5. Opérateurs, variables, expressions

- Reconnaissance d'un motif :
  - métacaractères : * ?
  - opérateurs : -like -notlike
    Exp. "mammia" –like '*ma*' → True

- Conformité à une expression régulière :
  - expressions : [abc] [a-z]
  - opérateurs : -match -notmatch

- Remplacement : -replace
  Exp. 'la bon confiture' –replace 'bon', 'bonne'

3. PowerShell

3.6. Construire un script Powershell

- Fichier .ps1
  ← reconnu par l'extension de fichier

- Vérifier les droits d'exécution
  → Get-ExecutionPolicy
  valeurs : Restricted, AllSigned, RemoteSigned, Unrestricted

- Si besoin, ajuster les droits
  → Set-ExecutionPolicy RemoteSigned

- Editer un script : Windows PowerShell ISE
  → clic-droit / modifier → un outil GUI complet

- Structures de contrôles classiques :
  - syntaxe proche de celle du C
  - en plus rigoureux

3. PowerShell

3.7. Structures de contrôle – Conditionnelle

Conditionnelle simple (SI-ALORS)

```
echo 'oui ?'
$x=read-host

if ($x -eq 'oui')
   { echo "d'accord" }
```

Conditionnelle complète (SI-ALORS-SINON)

```
echo 'oui ou non ?'
$x=read-host

if ($x -eq 'oui')
   { echo 'd'accord' }
else
   { echo 'pas d'accord' }
```


3. PowerShell

3.7. Structures de contrôle – Conditionnelle

Condition élaborée

```
echo 'lettre svp'

$x=read-host

if (($x -ge 'A') -and ($x -le 'Z'))
   { echo 'ok' }
else  { echo 'erreur' }
```

Conditionnelle multiple (SINON-SI)

```
if ...
elseif ...
...
else ...
```

N.B. pas d'alias read

3. PowerShell

3.7. Structures de contrôle – Branchement multiple

Branchement multiple sur une variable

```
$x=read-host

switch  ($x)
   {
   0   { echo 'valeur nulle' }
   1   { echo 'un' }
   2   { echo 'deux' }
   default  { echo 'je ne sais pas' }
   }
```

N B. Branchement multiples sur de nombreux objets ...

3. PowerShell

3.7. Structures de contrôle – Branchement multiple

Branchement multiple avec expression rationnelle

```
switch –regex ($x)
   {
   '^[0-9]'      { echo "$x commence par un chiffre" }
   '^[a-zA-Z]  { echo "$x commence par une lettre" }
```

Variante :

```
$x=read-host

switch –regex -casesensitive ($x)
   {
   '^[a-z]'   { echo "$x commence par une minuscule" }
   '^[A-Z]'   { echo "$x commence par une MAJUSCULE" }
   default  { echo 'bof!' }
   }
```


3. PowerShell

3.7. Structures de contrôle – Boucles

Boucle bornée (POUR)

```
$tableau=0..9

for ($i=0 ; $i –le 9 ; $i++)
   {
   echo $tableau[$i]
   }
```

Variante 1 : `for ($i=0 ; $i –le 9 ; $i++)  { echo $tableau[$i] }`

Variante 2 (déconseillée) :

```
for ($i=0; $i –le 9)   { echo $tableau[$i] ; $i++ }
```

N B. { } obligatoires
Possible : echo $tableau¶ ou simplement $tableau¶
echo → Write-Host

3. PowerShell

3.7. Structures de contrôle – Boucles

Boucle énumérative (POURCHAQUE)

```
$tableau=0..9

foreach ($i in $tableau)
   {
   echo $i
   }
```

N.B. foreach → Foreach-Object

3. PowerShell

3.7. Structures de contrôle – Boucles

Boucle précontrôlée (TANTQUE)

```
$tableau=0..9
$i=0

while ($i –lt 9)
   {
   write-host $tableau[$i]
   $i++
   }
```


3. PowerShell

3.7. Structures de contrôle – Boucles

Boucle postcontrôlée (REPETER)

```
$tableau=0. 9
$i=0

do
   {
   write-host $tableau[$i]
   $i++
   }
while ($i –lt 9)
```


3. PowerShell

3.8. Paramètres

Script donnerdate.ps1 :

```
if ($args.Length –lt 3)
   { echo "pas assez s'arguments" }
else
   { echo "nous sommes le "  $args[0]  $args[1]  $args[2] }
```

- Utilisation en ligne de commande :

```
donnerdate 1 avril 2016
```

Variante : echo "nous sommes le $args"

3. PowerShell

3.9. Fonctions

Fonction sans paramètre (procédure)

- Définir :

```
Function donnerdate
{
   echo 'année ?'
   $annee=read-host
   echo 'mois ?'
   $mois=read-host
   echo 'jour ?'
   $jour=read-host
   echo "nous sommes le $jour $mois $annee"
}
```

- Utiliser :

```
donnerdate
```

3. PowerShell
3.9. Fonctions

Fonction avec paramètre(s)

- Définir :

```
Function donnerdate
{
   param($jour, $mois, $annee)
   echo "nous sommes le $jour $mois $annee"
}
```

- Utiliser :

```
echo 'année ?' ; $a=read-host
echo 'mois ?' ; $m=read-host
echo 'jour ?' ; $j=read-host

donnerdate $j $m $a
```


3. PowerShell
3.9. Fonctions

Fonction avec retour explicite

- Définir :

```
Function donnerdate jour mois annee
{
   param($jour, $mois, $annee)
   return "nous sommes le $jour $mois $annee"
}
```

- Utiliser :

```
echo 'année ?' ; $a=read-host
echo 'mois ?' ;  $m=read-host
echo 'jour ?' ;  $j=read-host

donnerdate $j $m $a
```


3. PowerShell
3.9. Fonctions – Fonctions avancées

Filtrage (Exp. d'après Petitjean & Lemesle)

```
Filter filtrer-dossier
{ if ($_.mode –like "d*") { $_ } }

get-childitem c: | filtrer-dossier
```

```
Répertoire : C:\
Mode         LastWriteTime     Length   Name
----         -------------     ------   ----
...
d----        14/03/2016 11:01           opt
d-r--        27/02/2016 15:07           Program Files
d-r--        16/03/2016 10:08           Program Files (x86)
da---        15/09/2015 18:02           Users
da---        18/03/2016 06:50           Windows
...
```


3. PowerShell
3.10. Items et providers

Get-PSProvider

```
Name            Capabilities                    Drives
----            ------------                    ------
Alias           ShouldProcess                   {Alias}
Environment     ShouldProcess                   {Env}
FileSystem      Filter, ShouldProcess, Cre...   {C, D, E}
Function        ShouldProcess                   {Function}
Registry        ShouldProcess, Transactions     {HKLM, HKCU}
Variable        ShouldProcess                   {Variable}
```

get-command *item* | Filter prendre-item { if ($_.name -like "*item*") { $_ } }
| Get-command | prendre-item

CommandType	*Name*	*ModuleName*
Function	Get-DAEntryPointTableItem	DirectAcc...
...		
Cmdlet	Clear-Item	Microsoft...
...		

Get-ChildItem C: / Alias: / Env: / Function: / Variable: / Cert: / HKLM: / KHCU / ...

4. Fonctionnalités GUI de PowerShell

Exp.

```
Function popitup
{
    param ($message)
    $WshShell= new-object –comobject wscript.shell
    $WshShell.popup($message, 0, 'Mon popup')
}

popitup "bonjour chez vous !"
```

ComObject ← API Windows COM (*Component Object Model*)

5. Framework .NET

- Composante de Windows

- Bibliothèques de classes (objets) +Environnement d'exécution
  → utilisable avec PowerShell (compatible CLS :"Common Language Runtime")

  Exp.
  Objet : [system.dateTime]
  Fonction : Now
  Appel sous PowerShell : [system dateTime]::Now
  → dimanche 1 avril 2016 12:00:00

- Programmation GUI : interfaces fenêtrées (WinForm)
  → comparer avec programation X Window