Windows PowerShell 入門

Windows PowerShell に親しもう

コマンドレットのパイプ処理とは

システム理解のため実際に使ってみよう

Prepared by 遠藤忠雄

Windows PowerShell の起動

(Windows 10 の場合)

- スタートボタンを右クリックし現れたメニューの Windows PowerShell をクリック

（ または ）

- Windowsキー ＋ Xキーを押し、現れたメニューのWindows PowerShell をクリック

注）メニュー内にWindows PowerShellがない場合は「設定」、「個人用設定」、「タスクバー」内のトグルスイッチ「スタートボタンを右クリックするか Windowsキー ＋ X キーを押した時に表示されるメニューで・・・」をオンにする

コマンドウィンドウ（その1）

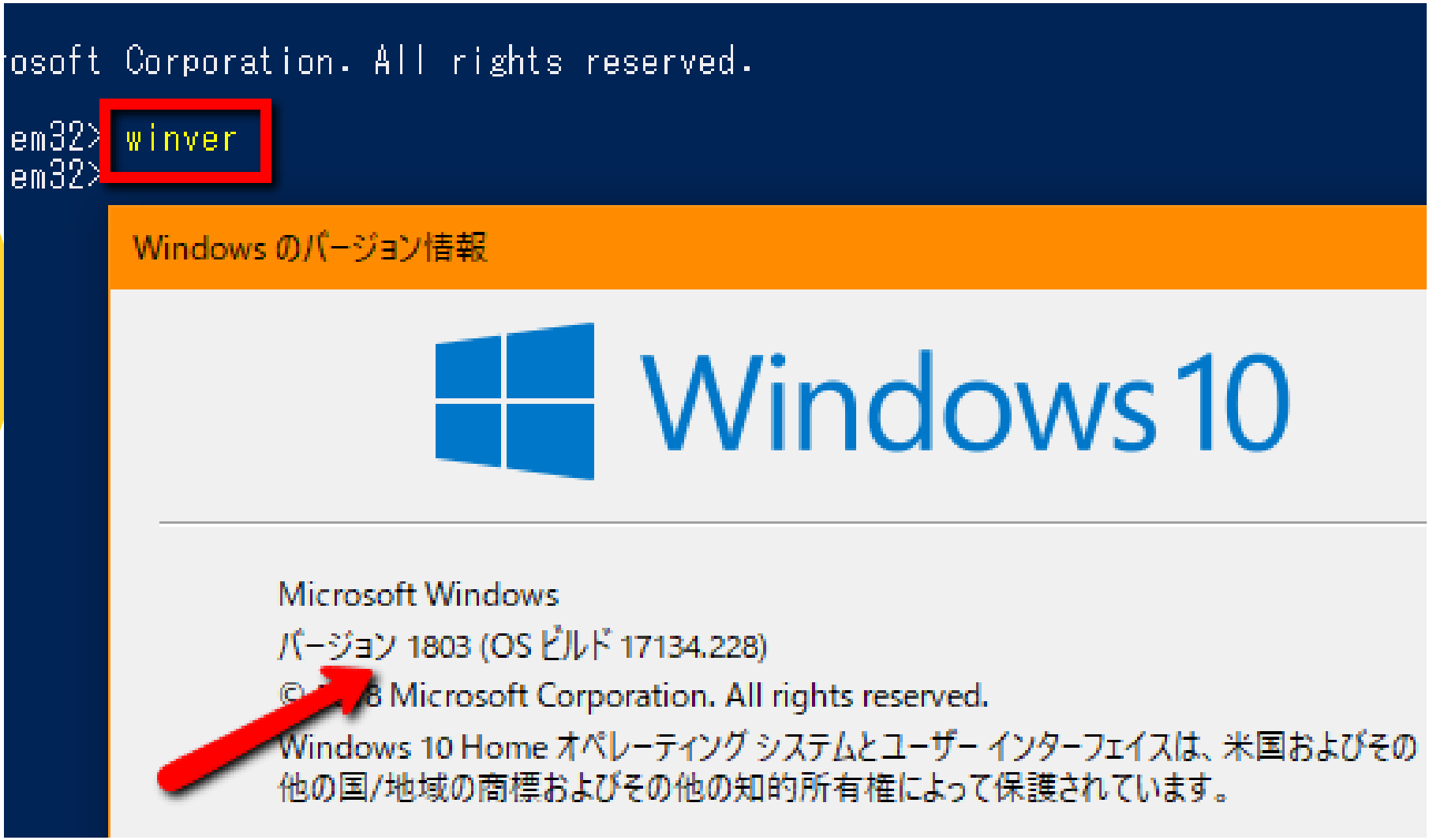
従来のMS-DOSコマンドも使用できる

```
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:¥WINDOWS¥system32> ipconfig

Windows IP 構成


イーサネット アダプター イーサネット:

   メディアの状態. . . . . . . . . . . . : メディアは接続されていません
   接続固有の DNS サフィックス . . . . . :

Wireless LAN adapter ローカル エリア接続* 2:

   メディアの状態. . . . . . . . . . . . : メディアは接続されていません
   接続固有の DNS サフィックス . . . . . :

Wireless LAN adapter ローカル エリア接続* 3:

   メディアの状態. . . . . . . . . . . . : メディアは
   接続固有の DNS サフィックス . . . . . :

イーサネット アダプター イーサネット 2:

   メディアの状態. . . . . . . . . . . . : メディアは
   接続固有の DNS サフィックス . . . . . :

Wireless LAN adapter Wi-Fi:

   接続固有の DNS サフィックス . . . . . : tschr1.kt.home.ne.jp
   リンクローカル IPv6 アドレス. . . . . : fe80::c0f9:65c1:1f2b:96e4%12
   IPv4 アドレス . . . . . . . . . . . . : 192.168.10.5
   サブネット マスク . . . . . . . . . . : 255.255.255.0
   デフォルト ゲートウェイ . . . . . . . : 192.168.10.254
```

IPCONFIG
コマンドの例
Wi-Fi 接続のIP
アドレスなどの
情報

コマンドウィンドウ（その2）

WinVerコマンドの例



コマンドレット（その１）
(Windows PowerShell 専用のコマンド)

- 命名形式は、動作-目的で統一されている
  例：Clear-History、Get-ChildItem など
- 入力は大文字小文字の区別はされない
- 文字数が多く入力しにくいので別名（エイリアス）が用意されている　例：ClHy、GCi など
- コマンドレットの数は多く、現在 544 個、別名の数は153個ある

注）1つのコマンドレットに複数の別名がある場合がある
例：Get-History にはGHy、HistoryおよびHの3つの別名がある

コマンドレット（その2）

（Get-Command 別名は gcm で実行）

```
PS C:¥WINDOWS¥system32> gcm

CommandType     Name                                               Version
-----------     ----                                               -------
Alias           Add-ProvisionedAppxPackage                         3.0
Alias           Add-ProvisioningPackage                            3.0
Alias           Add-TrustedProvisioningCertificate                 3.0
Alias           Apply-WindowsUnattend                              3.0
Alias           Disable-PhysicalDiskIndication                     2.0.0.0
Alias           Disable-StorageDiagnosticLog                       2.0.0.0
Alias           Enable-PhysicalDiskIndication                      2.0.0.0
Alias           Enable-StorageDiagnosticLog                        2.0.0.0
Alias           Flush-Volume                                       2.0.0.0
Alias           Get-DiskSNV                                        2.0.0.0
Alias           Get-PhysicalDiskSNV                                2.0.0.0
Alias           Get-ProvisionedAppxPackage                         3.0
Alias           Get-StorageEnclosureSNV                            2.0.0.0
Alias           Initialize-Volume                                  2.0.0.0
Alias           Move-SmbClient                                     2.0.0.0
Alias           Optimize-ProvisionedAppxPackages                   3.0
Alias           Remove-EtwTraceSession                             1.0.0.0
Alias           Remove-ProvisionedAppxPackage                      3.0
Alias           Remove-ProvisioningPackage                         3.0
Alias           Remove-TrustedProvisioningCertificate              3.0
Alias           Set-EtwTraceSession                                1.0.0.0
Alias           Write-FileSystemCache                              2.0.0.0
```

注）コマンドレットを含む非常に多くの全てのコマンドが表示され、後半がコマンドレット

コマンドレットのパイプ処理

（縦棒｜で区切って入力）

```
PS C:¥WINDOWS¥system32> gcm | Where-Object {$_.commandtype -eq "cmdlet"}

CommandType     Name                                               Version
-----------     ----                                               -------
Cmdlet          Add-AppxPackage                                    2.0.0.0
Cmdlet          Add-AppxProvisionedPackage                         3.0
Cmdlet          Add-AppxVolume                                     2.0.0.0
Cmdlet          Add-BitsFile                                       2.0.0.0
Cmdlet          Add-CertificateEnrollmentPolicyServer              1.0.0.0
Cmdlet          Add-Computer                                       3.1.0.0
Cmdlet          Add-Content                                        3.1.0.0
Cmdlet          Add-History                                        3.0.0.0
Cmdlet          Add-JobTrigger                                     1.1.0.0
Cmdlet          Add-KdsRootKey                                     1.0.0.0
Cmdlet          Add-LocalGroupMember                               1.0.0.0
Cmdlet          Add-Member                                         3.1.0.0
Cmdlet          Add-PSSnapin                                       3.0.0.0
Cmdlet          Add-Type                                           3.1.0.0
Cmdlet          Add-WindowsCapability                              3.0
```

注）前のコマンドレットの結果を次のコマンドレットに引き渡す

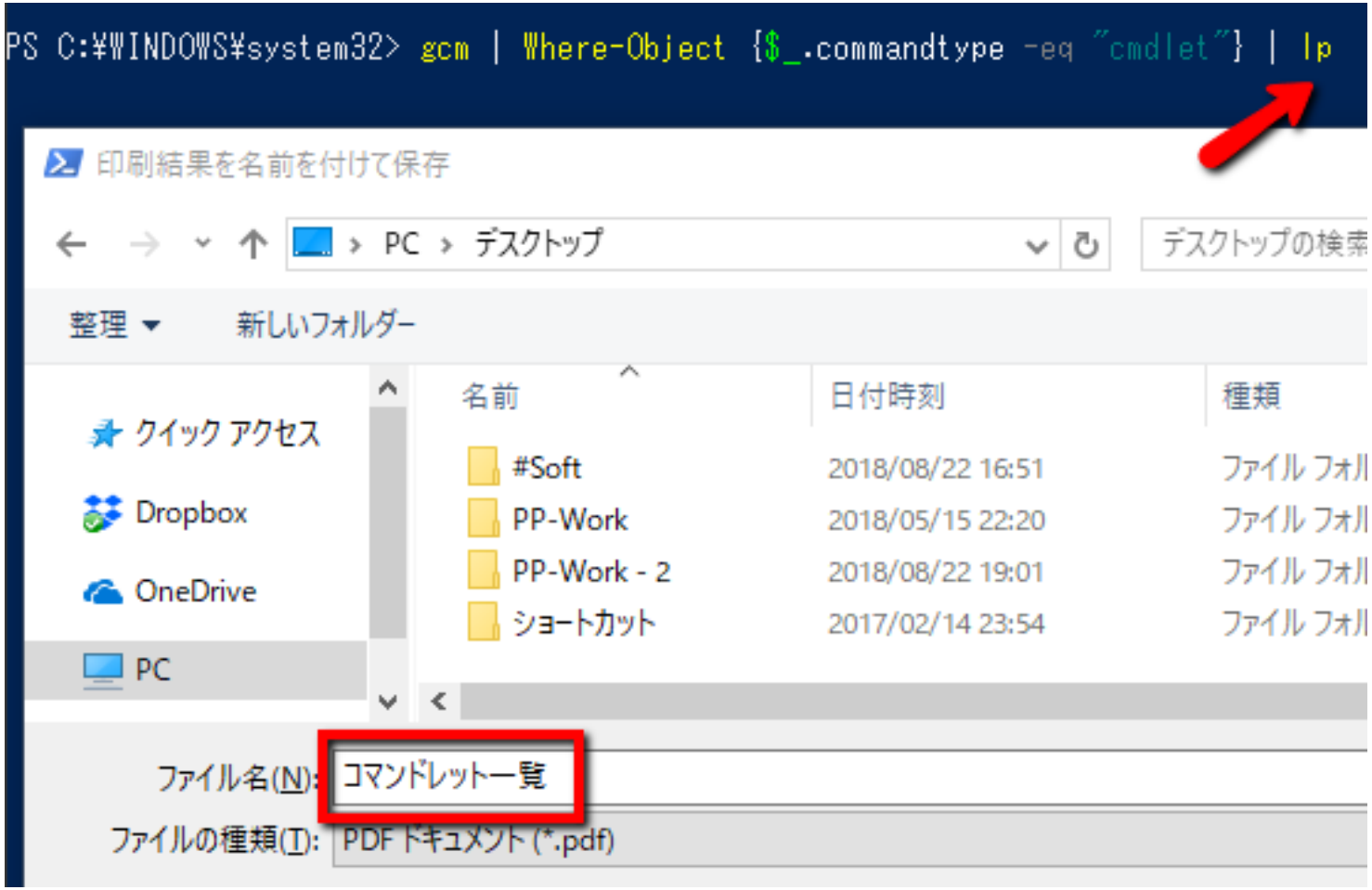
フォルダの作成
（New-Item コマンドレット、別名は ni で作成）

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:¥WINDOWS¥system32> ni c:¥work -type d


    ディレクトリ: C:¥


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        2018/08/24      9:33                work
```

注）Cドライブに work フォルダを作成、-type d の指定が必要

PDFファイルの作成（その1）

（Out-Printer 別名は LP ）

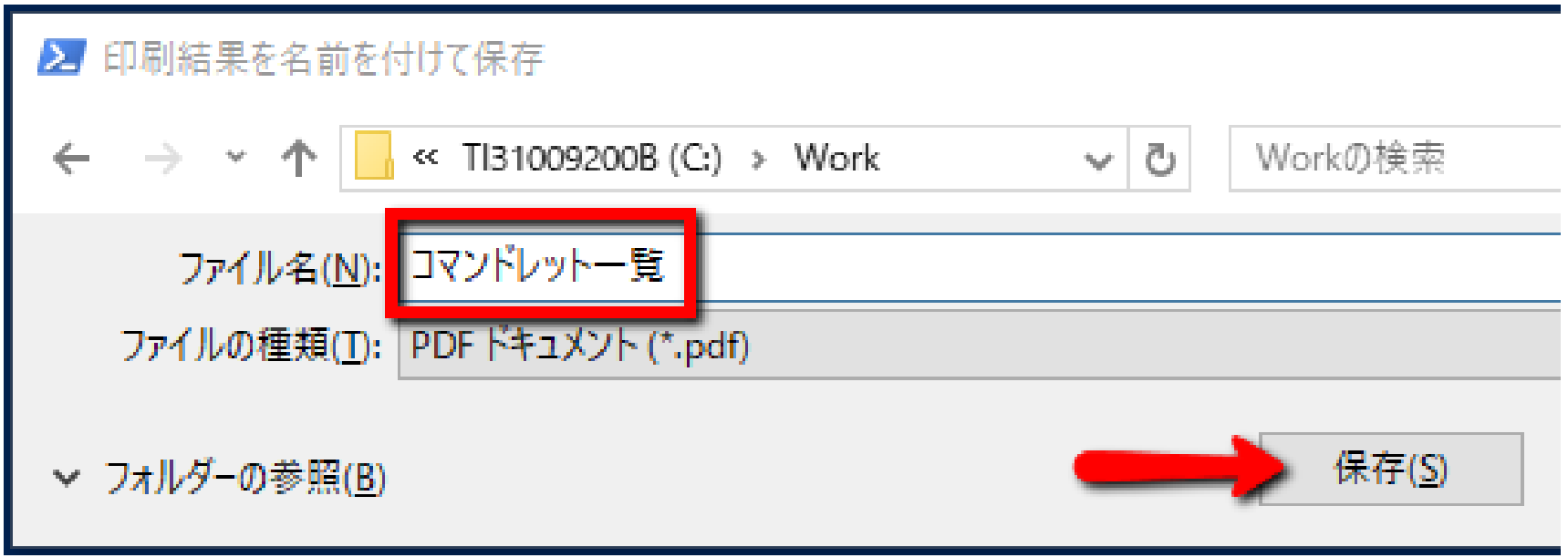


注）通常使うプリンターをMicrosoft Print to PDFにしておく

PDFファイルの作成（その2）

（lpコマンドレットにMicrosoft Print to PDF を指定 ）

```
gcm | where {$_.commandtype -eq "cmdlet"} | lp "Microsoft Print to PDF"
```



注）上の例はWorkフォルダ内にファイル名を入力して保存

コマンドレット（その2）

（Get-Command 別名は gcm で表示される）

```
gcm |where {$_.commandtype -eq "cmdlet"}|lp "microsoft print to pdf"
gcm |where {$_.commandtype -eq "cmdlet" -and $_.name -like "get-co*"}
```

```
CommandType     Name
-----------     ----
Cmdlet          Get-Command
Cmdlet          Get-ComputerInfo
Cmdlet          Get-ComputerRestorePoint
Cmdlet          Get-Content
Cmdlet          Get-ControlPanelItem
Cmdlet          Get-Counter
```

注）F8キーを利用し後半を打ち換え、Get-Coで始まるコマンドレットのみを表示、ワイルドカード＊は任意の文字列

効率的コマンド入力（その1）

（F8 キーの利用）

```
¥system32> gcm | where {$_.commandtype -eq "cmdlet" -and $_.name -like "get-com*"}

   Name                                               Version    Source
   ----                                               -------    ------
   Get-Command                                        3.0.0.0    Microsoft.PowerShe
   Get-ComputerInfo                                   3.1.0.0    Microsoft.PowerShe
   Get-ComputerRestorePoint                           3.1.0.0    Microsoft.PowerShe


¥system32> gcm | where {$_.commandtype -eq "cmdlet" -and $_.name -like "out*"}

   Name                                               Version    Source
   ----                                               -------    ------
   Out-Default                                        3.0.0.0    Microsoft.PowerShe
   Out-File                                           3.1.0.0    Microsoft.PowerShe
   Out-GridView                                       3.1.0.0    Microsoft.PowerShe
   Out-Host                                           3.0.0.0    Microsoft.PowerShe
   Out-Null                                           3.0.0.0    Microsoft.PowerShe
   Out-Printer                                        3.1.0.0    Microsoft.PowerShe
   Out-String                                         3.1.0.0    Microsoft.PowerShe
```

注）"get-com*"を"out*"に変更しコマンドを実行

ファイルの作成
（New-Item 別名は ni で作成）

```
PS C:¥WINDOWS¥system32> ni c:¥work¥test


    ディレクトリ: C:¥work


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        2018/08/24     10:17              0 test
```

注）Cドライブにある work フォルダ内に test ファイルを作成

システム復元ポイントの作成

（CheckPoint-Computerで作成「される、別名はない）

```
PS C:¥WINDOWS¥system32> CheckPoint-Computer

コマンド パイプライン位置 1 のコマンドレット Checkpoint-Computer
次のパラメーターに値を指定してください:
Description: Ver 1803
警告: 過去 1440
分以内に作成されたシステム復元ポイントが既に存在するため、新しいシステム
復元ポイントを作成できません。復元ポイントの作成頻度を変更するには、レジ
ストリ キー 'HKLM¥Software¥Microsoft¥Windows
NT¥CurrentVersion¥SystemRestore' の下に、DWORD 値
'SystemRestorePointCreationFrequency' を作成します。このレジストリ
キーの値で、復元ポイントを連続して作成する際に確保する必要がある間隔を分
単位で指定します。既定値は 1440 分 (24 時間) です。
PS C:¥WINDOWS¥system32>
```

注）通常復元ポイントの連続作成は24時間の経過が必要

システム復元ポイントの表示

（Get-ComputerRestorepoint で表示される、別名はない）

```
PS C:¥WINDOWS¥system32> get-computerRestorepoint

CreationTime              Description                          SequenceNumber
------------              -----------                          --------------
2018/05/08 15:57:43       Installed KAWAI バンドプロ...        1
2018/05/08 16:52:57       Windows バックアップ                 2


PS C:¥WINDOWS¥system32> get-computerRestorepoint

CreationTime              Description                          SequenceNumber
------------              -----------                          --------------
2018/05/08 15:57:43       Installed KAWAI バンドプロ...        1
2018/05/08 16:52:57       Windows バックアップ                 2
2018/05/09 21:59:16       Ver 1803                             3
```

注）上の例は5月9日21時59分16秒に復元ポイントを作成

日付と時刻又はどちらかの表示

（Get-Dateのパラメーターで指定、別名はない）

```
PS C:¥WINDOWS¥system32> Get-Date

2018年5月11日 13:50:19


PS C:¥WINDOWS¥system32> Get-Date -DisplayHint Date

2018年5月11日


PS C:¥WINDOWS¥system32> Get-Date -DisplayHint Time

13:52:20
```

今日から2020年大晦日までの日数

(New-TimeSpan で表示される、別名はない)

```
PS C:¥WINDOWS¥system32> r 8
New-TimeSpan $(Get-Date) $(Get-Date -month 12 -day 31 -year 2020)


Days              : 965
Hours             : 0
Minutes           : 0
Seconds           : 0
Milliseconds      : 0
Ticks             : 83376000000000
TotalDays         : 965
TotalHours        : 23160
TotalMinutes      : 1389600
TotalSeconds      : 83376000
TotalMilliseconds : 83376000000
```

注)上の例は2018年5月11日にコマンドレットを入力した結果

効率的コマンド入力(その2)

(Get-History,別名 H の利用)

```
PS C:¥WINDOWS¥system32> h

  Id CommandLine
  -- -----------
   1 ipconfig
   2 winver
   3 gcm
   4 ni c:¥work -type d
   5 gcm | where {$_.commandtype -eq "cmdlet"} |lp "Microsoft Print to PDF"
   6 gcm | where {$_.commandtype -eq "cmdlet" -and $_.name -like "get-co*"}
   7 gcm | where {$_.commandtype -eq "cmdlet" -and $_.name -like "out*"}
   8 ni c:¥work¥test
   9 CheckPoint-Computer
  10 grt-computerRestorepoint
  11 get-computerRestorepoint
  12 get-date
  13 new-timespan $(get-date) $(get-date -month 12 -day 31 -year 2020)
```

効率的コマンド入力(その3)

(Invoke-History別名　R の利用)

```
PS C:¥WINDOWS¥system32> r 7
gcm | where {$_.commandtype -eq "cmdlet" -and $_.name -like "out*"}

CommandType     Name                                               W
-----------     ----                                               -
Cmdlet          Out-Default                                        3
Cmdlet          Out-File                                           3
Cmdlet          Out-GridView                                       3
Cmdlet          Out-Host                                           3
Cmdlet          Out-Null                                           3
Cmdlet          Out-Printer                                        3
Cmdlet          Out-String                                         3
```

注)r 7で7番目の gcm | where コマンドレットを実行

入力履歴からの再実行

(Invoke-History で再実行)

```
PS C:¥WINDOWS¥system32> r 6
gcm | where {$_.commandtype -eq "cmdlet" -and $_.name -like "get-co*"}

CommandType     Name                                               Version
-----------     ----                                               -------
Cmdlet          Get-Command                                        3.0.0.0
Cmdlet          Get-ComputerInfo                                   3.1.0.0
Cmdlet          Get-ComputerRestorePoint                           3.1.0.0
Cmdlet          Get-Content                                        3.1.0.0
Cmdlet          Get-ControlPanelItem                               3.1.0.0
Cmdlet          Get-Counter                                        3.0.0.0
```

注1）上の例は履歴の6番目のコマンドを実行

注2）Invoke-History コマンドレットの最も簡単な別名は、R の1文字

プロセスとは（狭い意味で）

- 簡単に言うと、実行中のプログラムのこと
- 実行コードはメモリー上に読み込まれ
- CPUによって逐次取り出されて実行される
- 実行中となったプログラムの事をプロセスと言う
- タスクと同義で使われることが多い
- OSから見たプロセスはメモリーの管理単位である
- Get-Processコマンドレットで取得できる

全ての実行中のプロセス

（Get-Process で表示される、別名は gps ）

```
PS C:¥WINDOWS¥system32> Get-Process | where {$_.cpu -gt 1000}

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
    127       8     2164       7052   1,578.89   14616   0 dllhost
    270      15     3472      14808   1,304.05   16720   0 svchost
    305      16     3708      14964   1,386.03   16788   0 svchost
    341      29     4136      11656   6,606.63   17064   0 svchost
   4445       0      220      27864   1,164.98       4   0 System


PS C:¥WINDOWS¥system32>
```

注1）合計CPU時間が1,000（秒）を越えるプロセスのみ表示

注2） $_ はパイプ経由で渡されたオブジェクトを表す特殊変数

表示されるプロセスのプロパティ

Handles	プロセスがオープンしたハンドルの数
NPM(K)	プロセスが使用している非ページメモリのサイズ(KB)
PM(K)	プロセスが使用しているページメモリのサイズ(KB)
WS(K)	プロセスのワーキングセットのサイズ(KB)
CPU(s)	プロセスが使用したプロセッサ時間(秒)
Id	プロセスのプロセスID
SI	プロセスのセッションID
ProcessName	プロセスのプロセス名

取得したデータの並べ替え

（Sort-Object 別名は sort ）

```
Get-ChildItem c:¥Windows¥fonts|Where {$_.name -like "ms*"} | sort length
```

```
    ディレクトリ: C:¥windows¥fonts


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        2018/01/05     21:10         222632 MSUIGHUR.TTF
-a----        2018/01/05     21:10         230220 MSUIGHUB.TTF
-a----        2017/09/29     22:41         301876 msyi.ttf
-a----        2017/09/29     22:41        9214692 msgothic.ttc
-a----        2017/09/30     23:37       10081800 msmincho.ttc
-a----        2017/09/29     22:41       12139380 msyhl.ttc
-a----        2017/09/29     22:41       12875900 msjhl.ttc
-a----        2017/09/29     22:41       14439388 msjhbd.ttc
-a----        2017/09/29     22:41       16829116 msyhbd.ttc
-a----        2017/09/29     22:41       19647736 msyh.ttc
-a----        2017/09/29     22:41       21402208 msjh.ttc
```

注）上の例は ms で始まるFontファイルの昇順での並べ替え

3つのLogファイルの作成

（Out-File で新規ファイルを作成、別名はない）

```
Get-Process | Out-File c:¥work¥test1.log
Get-Process | Out-File c:¥work¥test2.log
Get-Process | Out-File c:¥work¥test3.log
$a = Get-Date; Add-Content c:¥work¥*.log $a
```

注1）上の例はGet-Processの結果を書き込んだtest1、test2およびtest3の3つLogファイルを作成し、その各々のファイルの最後にタイムスタンプを追加

注2） $a はGet-Dateの結果を入れるための変数

注3）¥*.log、は拡張子がlogの全てのファイル

タイムスタンプの追加

(Add-Content 別名 ac の優れた機能ワイルドカード文字)

```
PS C:¥WINDOWS¥system32> Get-Process | Out-File c:¥work¥test1.log
PS C:¥WINDOWS¥system32> Get-Process | Out-File c:¥work¥test2.log
PS C:¥WINDOWS¥system32> Get-Process | Out-File c:¥work¥test3.log
PS C:¥WINDOWS¥system32> $a = Get-Date; Add-Content c:¥work¥*.log $a
PS C:¥WINDOWS¥system32> H

  Id CommandLine
  -- -----------
   1 Get-Process | Out-File c:¥work¥test1.log
   2 Get-Process | Out-File c:¥work¥test2.log
   3 Get-Process | Out-File c:¥work¥test3.log
   4 $a = Get-Date; Add-Content c:¥work¥*.log $a


PS C:¥WINDOWS¥system32> R 4
$a = Get-Date; Add-Content c:¥work¥*.log $a
PS C:¥WINDOWS¥system32>
```

全てのLogファイルに2回目のタイムスタンプを追加

注)上の例は＊.logファイル (test1~3)全てにタイムスタンプを追加

フォルダ、ファイルの削除
（Remove-Item 別名は多い、ri rd erase rm rmdir del ）

```
PS C:¥WINDOWS¥system32> ri c:¥work -recurse
PS C:¥WINDOWS¥system32>
```

注1）Cドライブ内の work フォルダを削除、フォルダ内に子（ファイルやサブフォルダ）がある場合は –recurse を指定する

注2）–recurse を指定しない場合、確認が求められる

注3）このコマンドによる削除はごみ箱にも残らないので、重要なフォルダを間違って削除しないように特に要注意

復　習　問　題

Windows PowerShell（管理者）で実施する

1）使用しているPC（Wi-Fi接続）のIPアドレス及びOSのバージョンを調べる
2）現在のシステム復元ポイントを表示する
3）現在の日付と時刻を表示する
4）今日から2020年元日までの日数を調べる
5）続いてF8キーを利用し今日から2030年元日までの日数を調べる
6）合計CPU時間が500（秒）を越えるプロセスを表示する
7）ms で始まるFontファイルを取得し、Length の昇順で並べ替え表示する
8）コマンドレット一覧のPDFファイルをCドライブのWorkフォルダ内に作る
9）Get-Processの結果を書き込んだtestA、testBおよびtestCの３つLog
　ファイルをCドライブのWorkフォルダ内に作成し、その各々のファイルに
　現在のタイムスタンプを追加する（Workフォルダは前もって作成）
10）コマンドの入力履歴を表示する
11）入力履歴の再実行機能で上記 6）を再実行する
12）CドライブのWorkフォルダを削除する　　以上