PowerShell vs CMD: the Difference Explained

msp360.com/resources/blog/powershell-vs-cmd-the-difference-explained

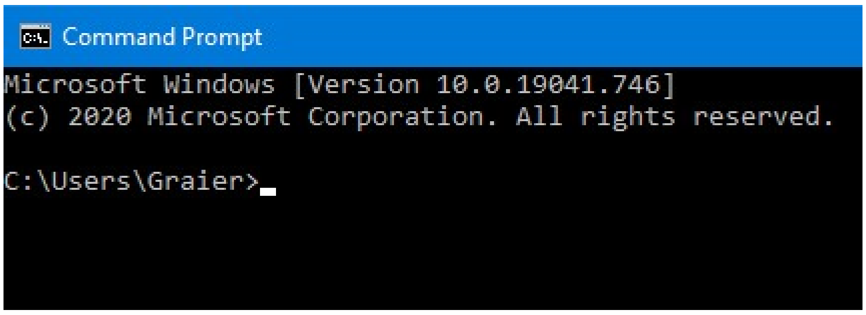
Ksenia Y

January 27, 2021

In Windows 10, the "traditional" command prompt that we've been using for years (dozens of years, to be exact) has been replaced by PowerShell. Of course, you can still call good old cmd.exe, but all menus and hotkeys now contain PowerShell as a default option, instead of CMD. Let's sort things out: why did this happen and what's the difference between these console applications?

Stay awhile and listen.

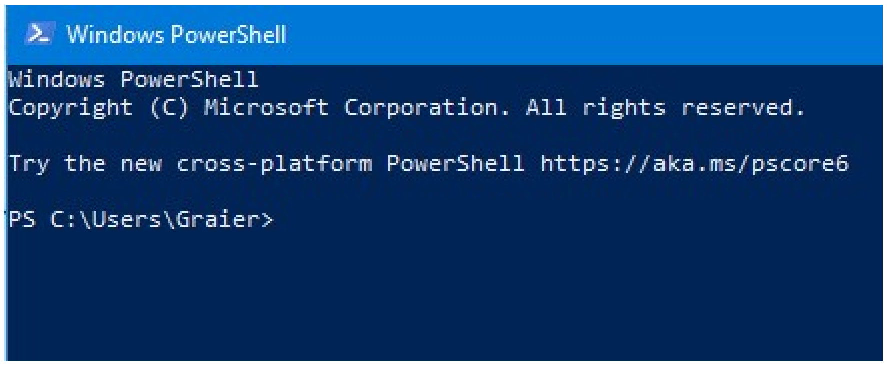
What Is CMD?



```
Microsoft Windows [Version 10.0.19041.746]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Graier>
```

CMD, or command prompt, originated from the Microsoft MS-DOS command-line shell and was used not only by system administrators but by end-users as well. (However, the dividing line between these two categories was, as you might remember, hardly noticeable.) Later on, when Microsoft introduced Windows NT, it was shipped with cmd.exe – the command line app that resembled its DOS predecessor and was compatible with it but also had additional functions. That was in 1987. Since then, cmd.exe has been a built-in app for all Windows operating systems.

What Is PowerShell?



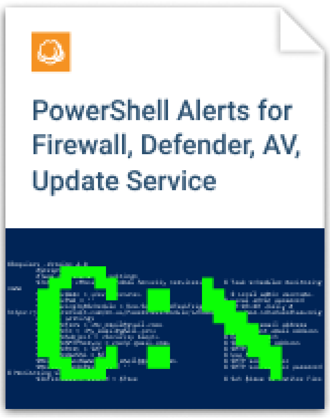
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Graier>
```

PowerShell is more than just a "powerful shell". After its release in 2006, this fully-fledged object-based scripting environment quickly became widespread among system administrators, who enjoyed its ease and functionality. PowerShell is based on CMD but leaves it far behind in terms of usefulness. With PowerShell, you can automate various tasks; for instance, managing Active Directory or implementing user access levels.

PowerShell Remote Alerts
In this document you'll find a PowerShell script that checks the status of the services listed below and sends an email alert if any of them is turned off:

- Windows Firewall
- Windows Defender
- Windows Update Service
- Any installed third-party antivirus

Download Now



PowerShell Alerts for Firewall, Defender, AV, Update Service

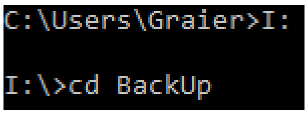
PowerShell vs Command Prompt: a Comparison

Before talking about the differences, it is worth discussing the characteristics that are shared by CMD and PowerShell. First of all, they have similar ancestors: the teletype machines of the 19th century. These machines were used in a similar way to all command lines nowadays. Even though technologies have developed drastically, a user still enters symbols on a machine and receives a result after these symbols have been interpreted and processed by this machine – like many years ago. Another point is that CMD and PowerShell appear to be attachments to the Windows console. Neither of them is a console itself; they're just additions.

And this functionality – we finally come to the PowerShell vs CMD technical comparison – differs a lot. Since CMD is a child of the MS-DOS command line, it inherited simple batch commands that any former DOS user remembers – like "cd" or "dir". These commands can be used to create scripts, and sometimes these scripts have to be very complicated in order to handle a given task. In PowerShell, there are no commands; instead, administrators utilize cmdlets ("command-lets") – small scripts with clear names. One PowerShell cmdlet can replace a long sequence of CMD commands. Here are some examples:

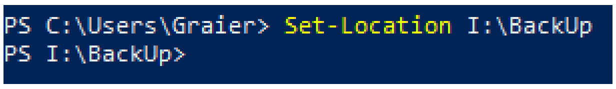
Changing a directory:

Command prompt



```
C:\Users\Graier>I:

I:\>cd BackUp
```

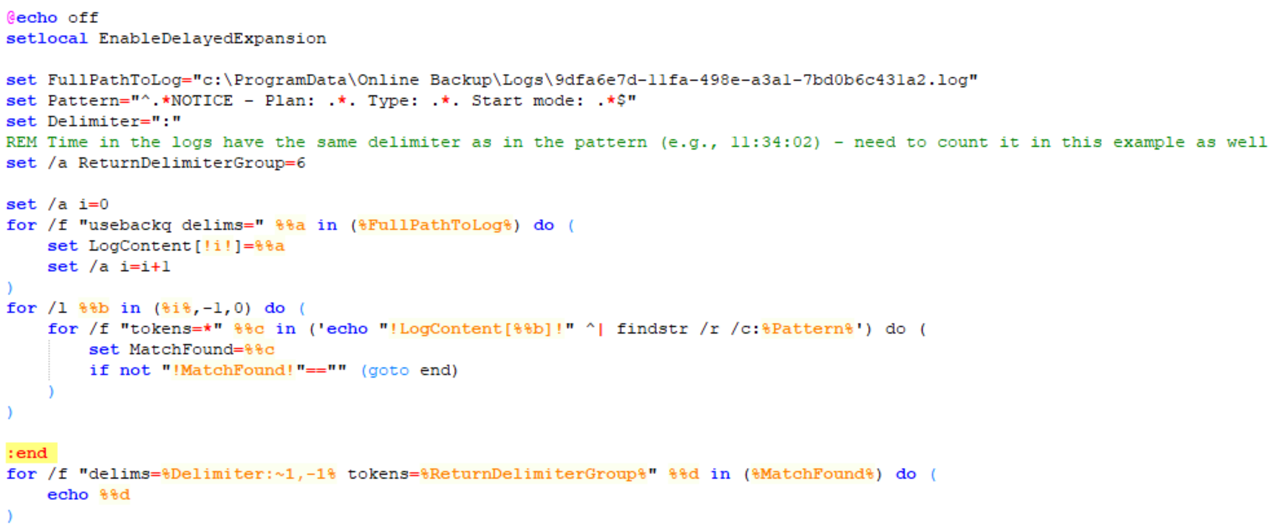
Windows PowerShell



```
PS C:\Users\Graier> Set-Location I:\BackUp
PS I:\BackUp>
```

Getting the last backup type from logs and displaying it

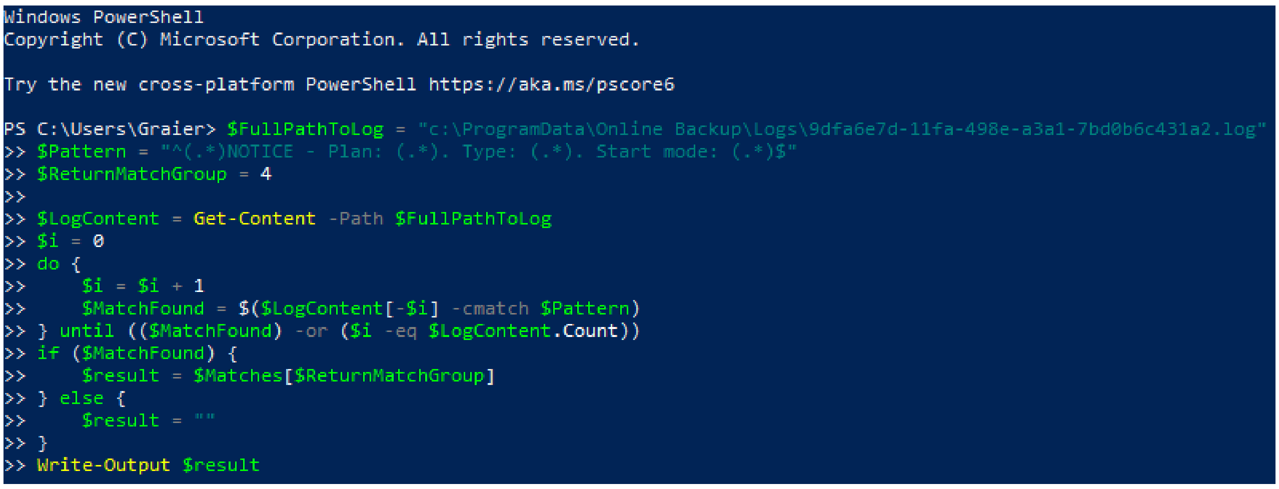
Command prompt script



```
@echo off
setlocal EnableDelayedExpansion

set FullPathToLog="c:\ProgramData\Online Backup\Logs\9dfa6e7d-11fa-498e-a3a1-7bd0b6c431a2.log"
set Pattern="^.*NOTICE - Plan: .*. Type: .*. Start mode: .*$"
set Delimiter=":"
REM Time in the logs have the same delimiter as in the pattern (e.g., 11:34:02) - need to count it in this example as well
set /a ReturnDelimiterGroup=6

set /a i=0
for /f "usebackq delims=" %%a in (%FullPathToLog%) do (
    set LogContent[!i!]=%%a
    set /a i=i+1
)
for /l %%b in (%i%,-1,0) do (
    for /f "tokens=*" %%c in ('echo "!LogContent[%%b]!" ^| findstr /r /c:%Pattern%') do (
        set MatchFound=%%c
        if not "!MatchFound!"=="" (goto end)
    )
)

:end
for /f "delims=%Delimiter:~1,-1% tokens=%ReturnDelimiterGroup%" %%d in (%MatchFound%) do (
    echo %%d
)
```

> Please note: this script doesn’t work from the command prompt itself, it should be launched as a .bat file

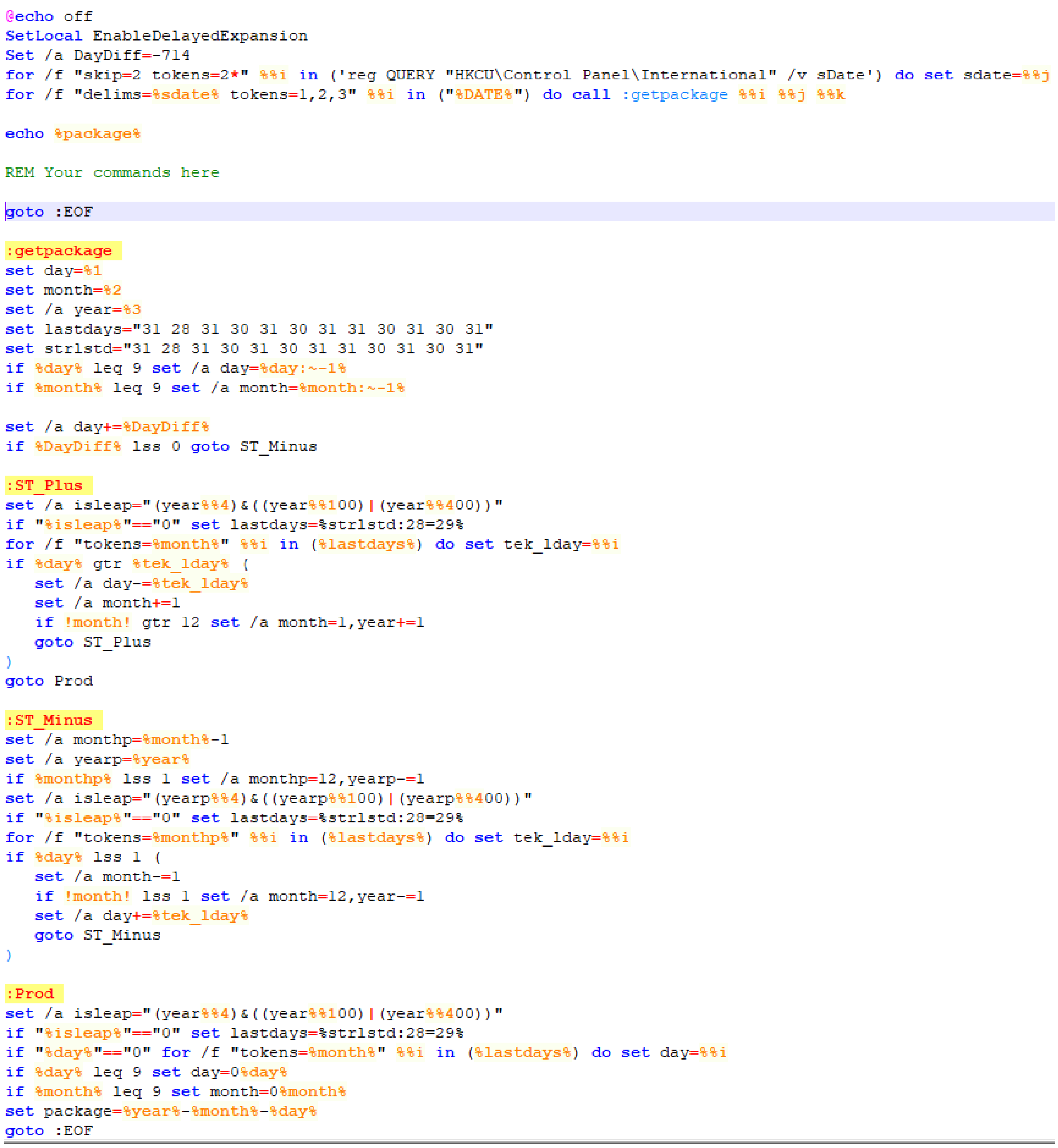
Windows PowerShell



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

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Graier> $FullPathToLog = "c:\ProgramData\Online Backup\Logs\9dfa6e7d-11fa-498e-a3a1-7bd0b6c431a2.log"
>> $Pattern = "^(.*)NOTICE - Plan: (.*). Type: (.*). Start mode: (.*)$"
>> $ReturnMatchGroup = 4
>>
>> $LogContent = Get-Content -Path $FullPathToLog
>> $i = 0
>> do {
>>     $i = $i + 1
>>     $MatchFound = $($LogContent[-$i] -cmatch $Pattern)
>> } until (($MatchFound) -or ($i -eq $LogContent.Count))
>> if ($MatchFound) {
>>     $result = $Matches[$ReturnMatchGroup]
>> } else {
>>     $result = ""
>> }
>> Write-Output $result
```

Calculating a date:

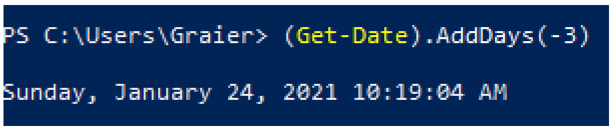
Command prompt script



```
@echo off
SetLocal EnableDelayedExpansion
Set /a DayDiff=-714
for /f "skip=2 tokens=2*" %%i in ('reg QUERY "HKCU\Control Panel\International" /v sDate') do set sdate=%%j
for /f "delims=%sdate% tokens=1,2,3" %%i in ("%DATE%") do call :getpackage %%i %%j %%k

echo %package%

REM Your commands here

goto :EOF

:getpackage
set day=%1
set month=%2
set /a year=%3
set lastdays="31 28 31 30 31 30 31 31 30 31 30 31"
set strlstd="31 28 31 30 31 30 31 31 30 31 30 31"
if %day% leq 9 set /a day=%day:~-1%
if %month% leq 9 set /a month=%month:~-1%

set /a day+=%DayDiff%
if %DayDiff% lss 0 goto ST_Minus

:ST_Plus
set /a isleap="(year%%4)&((year%%100)|(year%%400))"
if "%isleap%"=="0" set lastdays=%strlstd:28=29%
for /f "tokens=%month%" %%i in (%lastdays%) do set tek_lday=%%i
if %day% gtr %tek_lday% (
   set /a day-=%tek_lday%
   set /a month+=1
   if !month! gtr 12 set /a month=1,year+=1
   goto ST_Plus
)
goto Prod

:ST_Minus
set /a monthp=%month%-1
set /a yearp=%year%
if %monthp% lss 1 set /a monthp=12,yearp-=1
set /a isleap="(yearp%%4)&((yearp%%100)|(yearp%%400))"
if "%isleap%"=="0" set lastdays=%strlstd:28=29%
for /f "tokens=%monthp%" %%i in (%lastdays%) do set tek_lday=%%i
if %day% lss 1 (
   set /a month-=1
   if !month! lss 1 set /a month=12,year-=1
   set /a day+=%tek_lday%
   goto ST_Minus
)

:Prod
set /a isleap="(year%%4)&((year%%100)|(year%%400))"
if "%isleap%"=="0" set lastdays=%strlstd:28=29%
if "%day%"=="0" for /f "tokens=%month%" %%i in (%lastdays%) do set day=%%i
if %day% leq 9 set day=0%day%
if %month% leq 9 set month=0%month%
set package=%year%-%month%-%day%
goto :EOF
```

> Please note: this script doesn’t work from the command prompt itself, it should be launched as a .bat file

Windows PowerShell



```
PS C:\Users\Graier> (Get-Date).AddDays(-3)

Sunday, January 24, 2021 10:19:04 AM
```

Another difference is that CMD commands return text and, if a user needs some data, they must parse this text in order to get the required information. PowerShell cmdlets return objects that can be used for direct manipulation. To transfer these objects between cmdlets, pipes are used; they channel all the data so that it can be used in any number of cmdlets. Also, in CMD, the number of variables is limited and you have to pass them into commands in a strict order. PowerShell doesn’t have such limits.

Moreover, PowerShell is based on the .NET framework and can interact with any Windows objects, even core ones, unlike command prompt, which was not designed for system administration. While interacting with these objects, an administrator can create their own cmdlets in PowerShell to automate everyday tasks. In the process, administrators can use an embedded help system (it can be accessed via the Get-Help cmdlet) or test their creations. In the command prompt, there's no opportunity to test a script; you have to enter everything properly the first time or it won't work as expected and you may even lose some vital data. Thus, PowerShell is safer.

All this significantly improves the performance and usability of scripts; with PowerShell, you can do everything that CMD allows and much more. CMD is more backward-oriented; Microsoft wants it to be compatible with all old versions. So CMD might receive some updates, but not a lot. Accordingly, Microsoft switched to PowerShell, which gets regular updates and enhancements, and has a strong and active community.

However, note that you need to update PowerShell manually and, if your OS is old, there might be compatibility problems.

When to Use CMD

If you are used to CMD, of course, you can stay with it; it is still suitable for uncomplicated tasks. It is like a vintage car – no automation, no comfort, but you remember how helpful it was dozens of years ago and there's still some fuel in its tank. But what if you want to travel the highway?

When to Use PowerShell

PowerShell, with its wide functionality and steep learning curve, is the way to go. With it, you can ease your everyday tasks a lot, making scripts do all the work. Performing batch deployments, getting access to hidden data, managing permissions, and much more – PowerShell is great for any task related to the Windows OS.

Conclusion

As you can see, PowerShell and command prompt differ a lot, and this applies not only to their functionality but to their purposes as well. They might seem to be lookalikes but, in fact, PowerShell and CMD are like a space shuttle and a steam car.

Once extremely useful, the command prompt has now become a thing of the past. PowerShell completely replaces it and is able to process tasks that administrators of the 1980s couldn't even dream of. So, if you are still wondering which tool to use, there is a definite answer: PowerShell, for sure.

How to Remotely Manage Your Servers Using PSRemoting and Invoke PowerShell Commands

- Making a new firewall rule
- Restarting the server or computer

- Restating certain services
- Checking the status of the service, and more