PowerShell

Programmierung mit PowerShell
Fortschrittsbalken und Funktionen

Vorbereitung

Erstellen Sie für jede Aufgabe dieser Aufgabenserie ein eigenes PowerShell-Skript mit der Endung ps1. Führen Sie die Skripte aus, indem Sie mit einer PowerShell (ggf. als Administrator) in das Verzeichnis navigieren, in dem das Skript liegt.

Aufgabe 1

Implementieren Sie eine Funktion, die von 1 bis 50 zählt und den prozentualen Fortschritt beim Durchlaufen der Range als Fortschrittsbalken anzeigt. Der Fortschrittsbalken soll alle 100 Millisekunden aktualisiert werden, der Infotext soll *Suche…* sein.

Aufgabe 2

a. Implementieren Sie eine Funktion, die alle Dateien (keine Ordner!) im Verzeichnis der Umgebungsvariablen *$env:windir* ausgibt. Der Fortschritt beim Auflisten soll über einen Fortschrittsbalken angezeigt werden, der mitteilt, welche Datei gerade aufgelistet wird. Der Infotext soll *Finde Datei…* sein.

b. Unter GNU/Linux listen Sie alle Ordner (keine Dateien!) im Home-Verzeichnis des Benutzers auf. Der Fortschrittsbalken soll wie beschrieben funktionieren.

Aufgabe 3

Implementieren Sie die folgenden Funktionen mit PowerShell am Prompt oder in einer Skriptdatei. Der Eingabewert kann dabei zunächst eine Variable im Skript sein. Erweitern Sie Ihr Programm dann dahingehend, dass ein Parameter von der PowerShell übernommen wird, der den Eingabewert für die Funktion darstellt.

a. Betragsfunktion[1] *abs*

b. Signumfunktion[2] *sgn*

c. Fakultätsfunktion[3] *fac*

d. (Bonus) Binomialkoeffizient[4] *bk*

e. (Bonus) Ackermannfunktion[5] *ack*

---

[1] https://de.wikipedia.org/wiki/Betragsfunktion

[2] https://de.wikipedia.org/wiki/Vorzeichenfunktion

[3] https://de.wikipedia.org/wiki/Fakult%C3%A4t_(Mathematik)

[4] https://de.wikipedia.org/wiki/Binomialkoeffizient

[5] https://de.wikipedia.org/wiki/Ackermannfunktion

Aufgabe 4

a. Erstellen Sie eine ps1-Datei mit einer Funktion *Delete-FileTreeItems*, die alle Dateien in einem Dateisystembaum löscht (lassen Sie die Dateien hier zunächst nur auflisten anstatt sie tatsächlich zu löschen). Die Ordnerstruktur soll intakt bleiben, d.h. es wird kein Ordner (auch kein leerer) gelöscht.

b. Fügen Sie Ihrer Funktion einen Parameter *$root* vom Typ *string* hinzu, der angibt, wo die Wurzel des zu bearbeitenden Baumes liegt.

Aufgabe 5

Erstellen Sie eine Datei *GroupMembership.ps1* mit einer Funktion *Get-GroupMemberships*, die einen Benutzernamen als Parameter erhält und alle (lokalen) Gruppen anzeigt, in denen der Benutzer Mitglied ist.

Aufgabe 6

a. Erstellen Sie ein Skript *MeineFunktionen.ps1* und lassen Sie beim Ausführen den Text *Meine Funktionen werden geladen* ausgeben. Testen Sie nach jedem der folgenden Schritte die erstellen Funktion mit einigen Eingaben.

b. Fügen Sie eine neue Funktion *Get-MyProcess* hinzu, die Prozesse auf Ihrem Rechner, die mit dem Buchstaben ‚s' anzeigt. Die Ausgabe soll nur die Spalten *ID*, *ProcessName* und *CPU* beinhalten. Die Ausgabe soll nach der Property *CPU* absteigend sortiert werden.

c. Kopieren Sie die Funktion und fügen Sie unter dem Namen *Get-MyProcess2* eine weitere Version in das Skript ein. Wandeln Sie die Funktion wie folgt ab:

   - Die Funktion erhält einen Parameter *$FilterName*, der bestimmt, welche Prozesse angezeigt werden. Standardwert: Prozesse, die mit dem Buchstaben ‚s' beginnen.
   - Über einen Switch-Parameter *$Liste* wird gesteuert, ob die Ausgabe als Liste statt als Tabelle erfolgt.

d. Erstellen Sie eine Funktion *Get-MyProcess3*, die die folgenden Vorgaben erfüllt:

   - Vor der Verarbeitung soll das aktuelle Datum ausgegeben werden.
   - Dann soll der Code von *Get-MyProcess2* ausgeführt werden.
   - Nach der Verarbeitung soll der Text Aufgabe erledigt in schwarzer Schrift auf weißem Hintergrund ausgegeben werden.

e. Erstellen Sie eine kurze Dokumentation über die Funktion *Get-MyProcess2* und fügen Sie diese der Funktionsdefinition im Skript hinzu.

PowerShell

ANR

Programmierung mit PowerShell
Fortschrittsbalken und Funktionen

Aufgabe 7 (Bonus)

Nur unter GNU/Linux durchführbar.

Erstellen Sie eine Funktion *New-LocalLinuxUser*, die einen Benutzer in einer Linux-PowerShell anlegt. Die Funktion soll zwei Parameter vom Typ *string* erhalten, die für den Benutzernamen und das Passwort.

Aufgabe 8 (Bonus)

Nur unter GNU/Linux durchführbar.

Erstellen Sie eine Funktion Upgrade-System, die die Paketquellen aktualisiert und alle Pakete aktualisiert, für die neue Versionen in den Paketquellen verfügbar sind.

Ergänzen Sie Ihre Funktion um einen Switch-Parameter *-Force*, der auch zurückgehaltene Pakete zwangsweise aktualisiert.