HOWTO zum Aufruf von Shell-Skripten

(C) 2006-2024 T.Birnthaler/H.Gottschalk <howtos(at)ostc.de>
OSTC Open Source Training and Consulting GmbH
http://www.ostc.de

$Id: shell-call-HOWTO.txt,v 1.15 2025/02/18 10:08:31 tsbirn Exp $

Dieses Dokument beschreibt die verschiedenen Verfahren zum Aufruf von Shell-Skripten.

INHALTSVERZEICHNIS

1) Aufrufarten

---

1) Aufrufarten

Für den Aufruf eines Shell-Skriptes gibt es folgende Möglichkeiten, je nach Anwendungszweck ist eine geeignete auszuwählen:

A U F R U F	"PATH" durch-sucht	"x"-Recht nötig	Sub-Shell	Neuer Proz.	Env Var vererbt	Sh Var vererbt	Aliase+ Funkt. vererbt	Rückkehr zum Aufrufer
cmd.sh	JA	JA	JA	JA	J	-	--	JA
./cmd.sh	--	JA	JA	JA	J	-	--	JA
/PFAD/ZU/cmd.sh	--	JA	JA	JA	J	-	--	JA
bash [OPT] cmd.sh	--	--	JA	JA	J	-	--	JA
. cmd.sh	JA	--	--	--	J	J	JA	JA
source cmd.sh	JA	--	--	--	J	J	JA	JA
. ./cmd.sh	--	--	--	--	J	J	JA	JA
source ./cmd.sh	--	--	--	--	J	J	JA	JA
. /PFAD/ZU/cmd.sh	--	--	--	--	J	J	JA	JA
source /PFAD/ZU/cmd.sh	--	--	--	--	J	J	JA	JA
exec cmd.sh	JA	JA	--	--	J	-	--	--
exec ./cmd.sh	--	JA	--	--	J	-	--	--
exec /PFAD/ZU/cmd.sh	--	JA	--	--	J	-	--	--

* Das Kommando "." heißt auch "source"- oder "dot"-Kommando. Es liest die angegebene Datei in der aktuellen Shell ein (eine Art include-Anweisung), startet also KEINE Sub-Shell. Der Suchpfad wird dabei wie bei einem normalen Kommando-Aufruf durchsucht.

* Der Name von Shell-Skripten ist beliebig wählbar. Per Konvention wird aber meist die Extension ".sh" angehängt (ist beim Aufruf auch anzugeben!).

* Shell-Skripte müssen immer das Lese-Recht ("r"-Bit) gesetzt haben. Sofern sie als Kommando (ohne "sh", ".", "source", "exec" davor) aufgerufen werden sollen, müssen sie auch ausführbar sein ("x"-Bit) und in einem Verzeichnis im Suchpfad $PATH liegen.

* Der Suchpfad $PATH wird immer dann durchsucht, wenn nur ein KOMMANDO-NAME angegeben wurde. Da aus Sicherheitsgründen das AKTUELLE VERZEICHNiS "." nicht im Suchpfad steht, muss man Skripten im aktuellen Verzeichnis immer per "./SKRIPT" aufrufen.

* Nur EXPORTIERTE Umgebungs-Variablen (Environment Variables) werden an das aufgerufene Skript VERERBT (Befehl "export"). Shell-Variablen

sowie Aliase und Funktionen werden NICHT VERERBT.

* Eine in einem Eltern-Prozess EXPORTIERTE Variable ist auch im Kind-Prozesse EXPORTIERT. Will man das verhindern, muss der Kindprozess sie löschen. Ein Shell-Skript kann das durch "undef VARIABLE" erreichen.

* Wird eine Sub-Shell gestartet (Standardfall), dann können Variablen und sonstige Einstellungen aus dem gestarteten Skript NICHT an das aufrufende Skript zurückgegeben werden (die alten Werte in der Ausgangs-Shell bleiben erhalten). Dies wirkt auf den ersten Blick widersinning, ist allerdings unter dem Aspekt der Unabhängigkeit von Skripten ein gutes Konzept.

* Das Standardverhalten von DOS/Windows, alle Skripte im gleichen Umgebungsbereich auszuführen, führt dort zu vielen Problemen (Speicherüberlauf, Variablen überschrieben, Variablen unabsichtlich gesetzt, ...)

* Ein Aufruf über "exec" spart einen Prozeß ein, eine Rückkehr zum aufrufenden Prozeß ist allerdings nicht mehr möglich. Ist dann sinnvoll, wenn sehr viele Prozesse gestartet werden oder wenn der aufgerufene Befehl der letzte im Skript ist.

* Bei Shell-Skripten ist das SUID/SGID-Bit nicht wirksam, da sie nicht wie ein Binärprogram direkt ausführbar sind, sondern von einer Shell interpretiert werden.

* Unter LINUX sind aus diesem Grund beide Bits für Shell-Skripte nicht setzbar. Bei jeder Änderung an ausführbaren Dateien werden zudem diese beiden Bits aus Sicherheitsgründen zurückgesetzt und müssen explizit wieder gesetzt werden.