(C) 2024 OSTC GmbH (http://www.ostc.de)

HOWTO zum Aufruf von Shell-Skripten

(C) 2006-2017 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 1.14 2019/11/26 19:37:07 tsbirn Exp $

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

INHALTSVERZEICHNIS

1) Aufrufarten

---

1) Aufrufarten

FÃ¼r den Aufruf eines Shell-Skriptes gibt es folgende MÃ¶glichkeiten, je nach
Anwendungszweck ist eine geeignete auszuwÃ¤hlen:

Aufruf	"PATH" durchsucht	"x"-Recht nÃ¶tig	Sub-Shell	Neuer Proz.	Env Var vererbt	Sh Var vererbt	Aliase+Funkt. vererbt	RÃ¼ckkehr 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
sh [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
exec cmd.sh	Ja	Ja	--	--	J	-	--	--
exec ./cmd.sh	--	Ja	--	--	J	-	--	--

* Das Kommando "." heiÃM-^_t 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 wÃ¤hlbar. Per Konvention wird aber
  meist die Extension ".sh" angehÃ¤ngt (die dann beim auch Aufruf anzugeben ist!).

* Shell-Skripte mÃ¼ssen immer das Lese-Recht ("r"-Bit) gesetzt haben. Sofern
  sie als Kommando (ohne "sh", ".", "source", "exec" davor) aufgerufen werden
  sollen, mÃ¼ssen sie auch ausfÃ¼hrbar sein ("x"-Bit) und in einem Verzeichnis
  im Suchpfad $PATH liegen.

* Der Suchpfad $PATH wird immer dann durchsucht, wenn nur ein Kommandoname
  angegeben wurde. Da inzwischen bei allen Benutzern (insbesondere der
  "root"!) aus SicherheitsgrÃ¼nden das aktuelle Verzeichnis "." im Pfad fehlt,
  muss man Skripten im aktuellen Verzeichnis immer per "./SKRIPT" aufrufen.

* Von den Shell-Variablen werden nur die EXPORTIERTEN "Umgebungs-Variablen"
  an das aufgerufene Skript vererbt (Befehl "export"). "Shell-Variablen"
  sowie Aliase und Funktionen werden NICHT an Shell-Skripte vererbt.

* Wird eine Sub-Shell gestartet (Standardfall), dann kÃ¶nnen Variablen und
  sonstige Einstellungen aus dem gestarteten Skript NICHT an das aufrufende
  Skript zurÃ¼ckgegeben werden (die alten Werte in der Ausgangs-Shell bleiben
  erhalten). Dies wirkt auf den ersten Blick widersinning, ist allerdings
  unter dem Aspekt der UnabhÃ¤ngigkeit von Skripten ein gutes Konzept.

* Das Standardverhalten von DOS/Windows, alle Skripten im gleichen Umgebungs-
  bereich auszufÃ¼hren, hat dort zu vielen Problemen gefÃ¼hrt (SpeicherÃ¼berlauf,
  Variablen Ã¼berschrieben, Variablen unabsichtlich gesetzt, ...)

* Ein Aufruf Ã¼ber "exec" spart einen ProzeÃM-^_ ein, eine RÃ¼ckkehr zum aufrufenden
  ProzeÃM-^_ ist allerdings nicht mehr mÃ¶glich. Ist dann sinnvoll, wenn sehr viele
  Prozesse gestartet werden oder wenn der aufgerufenen Befehl der letzte im
  Skript ist.

* Shell-Skripte sollten kein SUID- oder SGID-Bit gesetzt haben, da sie immer das
  Lese-Recht gesetzt haben mÃ¼ssen und daher evtl. vorhandene SicherheitslÃ¶cher

(C) 2024 OSTC GmbH (http://www.ostc.de)

sehr einfach durch Lesen des Skriptes zu entdecken sind.

* Unter LINUX sind aus diesem Grund beide Bits fÃ¼r Shell-Skripten nicht setzbar.
  Bei jeder Ã M-^Dnderung an ausfÃ¼hrbaren Dateien werden zudem diese beiden Bits aus
  SicherheitsgrÃ¼nden zurÃ¼ckgesetzt und mÃ¼ssen explizit wieder gesetzt werden.