

Oct 06, 09 20:29

shell-shee-bang-HOWTO.txt

Page 1/2

HOWTO zum Shell-Shee-Bang-Zeile
 (C) 2006 T.Birnthaler/H.Gottschalk <howtos(at)ostc.de>
 OSTC GmbH, <http://www.ostc.de>
 \$Id: shell-shee-bang-HOWTO.txt,v 1.3 2008-12-22 14:37:47 tsbirn Exp \$

Dieses Dokument beschreibt die Bedeutung der Shell-Shee-Bang-Zeile.

Inhaltsverzeichnis

- 1) Einführung
- 2) Beispiele

1) Einführung

Die Shee-Bang-Zeile bestimmt, mit welchem Programm (Interpreter) eine Skript-Datei ausgeführt wird. Hierbei ist in der 1. Zeile ab der 1. Spalte der vollständige absolute Pfad zu dem Programm anzugeben und davor "#!" zu setzen. Sobald die Skript-Datei ausführbar gemacht wurde, ist sie unter ihrem Namen als Kommando aufrufbar.

Neben den klassischen Shell-Skripten für "sh", "csh", "tcsh", "ksh" und "bash" gibt es noch viele anderen Skript-Programmiersprachen (z.B. "sed", "awk", "perl", "php") die auf diese Weise zu einem ausführbaren Programm gemacht werden können.

Ein (klassisches) Bourne-Shell-Skript "skript.sh" beginnt mit "#!/bin/sh".

```
#!/bin/sh
echo "Hallo Welt!"
```

Ein Bash-Shell-Skript "bash.sh" (die Bash kann mehr als die Bourne-Shell) beginnt mit "#!/bin/bash" (oder "#!/usr/bin/bash"):

```
#!/bin/bash
#!/bin/sh (hat keine Wirkung, weil nicht in 1. Zeile!)
echo "Hallo Welt!"
```

Bei ungültigem Pfad (z.B. führenden "/" vergessen) erfolgt beim Aufruf des Skriptes die Fehlermeldung "bash: ./tippfelher.sh: bin/sh: bad interpreter: Datei oder Verzeichnis nicht gefunden":

```
#!/xyzabc123                # Kommando "xyzabc123" gibt es nicht
echo "Hallo Welt!"

#!/bin/sh                   # Pfad "bin/sh" gibt es nicht
echo "Hallo Welt!"
```

2) Beispiele

Ein kleines Perl-Skript "hello.pl" (-w schaltet die Warnungen ein):

```
#!/usr/bin/perl -w
print "Hallo Welt!\n";
```

Ein grep-Skript "made.grep", das Kommando "#!/bin/grep made" durchsucht den Inhalt des Skriptes nach Zeilen, die das Wort "made" enthalten (nicht in Anführungszeichen setzen, da diese sonst mitgesucht werden; die Shell sieht diese Kommandozeile nämlich gar nicht, sondern nur der Kern!):

```
#!/bin/grep made
echo "Dies ist ein Text mit made"
ohne ... ohne
mit made mit made mit made
ohne
ohne
```

Ein sed-Skript "vokal.sed" (sed = stream editor, programmierbarer Editor) zum automatisierten Bearbeiten von Dateien. Die Option "-f" (file) ist notwendig, damit der Sed-Interpreter das Skript aus einer Datei liest.

Aufruf: vokal.sed < DATEI > ERGEBNIS

```
#!/bin/sed -f
# Ersetzt alle Vokale durch nichts (s=substitute, g=global)
s/a//g
s/e//g
s/i//g
s/o//g
```

```
s/u//g
s/A//g
s/E//g
s/I//g
s/O//g
s/U//g
```

Ein awk-Skript "wc.awk" (awk = Vorläufer von Perl), das den Linux-Befehl "wc" (word count) nachprogrammiert. Der Pfad zum Awk-Interpreter kann unterschiedlich lauten (z.B. auch "/usr/bin/gawk"), die Option "-f" (file) ist notwendig, damit der Awk-Interpreter das Skript aus einer Datei liest.

```
#!/usr/bin/awk -f
# Zaehlt Zeilen, Worte und Zeichen einer Datei analog "wc"
{
    words += NF
    chars += length($0)
}
END {
    print NR, words, chars+NR
}
```