HOWTO zum Shell-Filename Globbing

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

$Id: shell-globbing-HOWTO.txt,v 1.14 2025/02/18 10:08:31 tsbirn Exp $

Dieses Dokument beschreibt die Shell-Dateinamen-Expansion ("Filename Globbing").

INHALTSVERZEICHNIS

1) Filename Globbing
2) Syntax der Suchmuster
3) Tipps
4) Beispiele

---

1) Filename Globbing

Um an ein Kommando eine Liste von Dateinamen zur Verarbeitung zu übergeben, können diese einzeln angegeben werden. Oder man gibt "Suchmuster" (Pattern) vor, die von der Shell in passende Dateinamen umgewandelt werden.

Die Shell expandiert zuerst alle Suchmuster zu einer Liste von passenden Dateinamen ("filename globbing"), setzt diese dann anstelle der Suchmuster ein und führt schließlich das Kommando aus (d.h. das Kommando "sieht" die Suchmuster gar nicht!).

Unabhängig vom auszuführenden Kommando wird die Dateinamen-Expansion immer von der Shell durchgeführt, BEVOR ein Kommando gestartet wird. Außer der Shell befasst sich also kein Kommando mit dem Aspekt der Dateinamen-Expansion, diese Funktionalität ist somit EINHEITLICH für alle Kommandos in der Shell realisiert.

2) Syntax der Suchmuster

Folgende Standard-"Metazeichen" sind zur Angabe von Suchmustern verwendbar:

Metazeichen	Bedeutung
*	0 ODER MEHR BELIEBIGE Zeichen (AUSSER Verz.trenner "/")
?	GENAU 1 BELIEBIGES Zeichen (AUSSER Verz.trenner "/")
[abc] [a-z]	GENAU 1 Zeichen aus Liste/Bereich (Zeichenmenge)
[!abc] [!a-z]	GENAU 1 Zeichen NICHT aus Liste/Bereich (sh, bash, ksh)
[^abc] [^a-z]	GENAU 1 Zeichen NICHT aus Liste/Bereich (csh, bash, ksh)
~	Home-Verzeichnis des aktuellen Benutzers
~USER	Home-Verzeichnis des Benutzers USER
{abc,def,...}	GENAU 1 der angegebenen Zeichenketten (csh, bash, ksh)
/	Verzeichnistrenner "/"
\C	Metazeichen C selbst (Backslash quotiert nächstes Zeichen)
\\	Zeichen "\" selbst

Folgende erweiterten Formen der "Metazeichen" gibt es:

**	0 oder mehr beliebige Zeichen (INKL. Verz.trenner "/")
**/	Pfad ohne Dateiname (INKL. Verz.trenner)

```
  |  ?(...|...|.) | 0/1 Vorkommen der Elemente in Musterliste ...  (bash, ksh) |
  |  *(...|...|.) | 0 ODER MEHR Vorkommen der Elemente ...         (bash, ksh) |
  |  +(...|...|.) | 1 ODER MEHR Vorkommen der Elemente ...         (bash, ksh) |
  |  @(...|...|.) | EINES der Elemente in Musterliste ...          (bash, ksh) |
  |  !(...|...|.) | ALLES AUSSER Elemente der Musterliste ...      (bash, ksh) |
  +--------------+-------------------------------------------------------------+
```

Diese Metazeichen können zu beliebig komplexen Suchmustern zusammengesetzt werden, passende Datei- und/oder Verzeichnisnamen müssen das Suchmuster VOLLSTÄNDIG erfüllen. Eine Längenbeschränkung oder eine Beschränkung in der Anzahl der verwendeten Metazeichen gibt es nicht.

```
  ls *-{jan,feb,mar,apr,may,jun,jul,aug,sep,oct,nov,dec}-*   # -MMM-
  ls [a-z][a-zA-Z0-9_]*.txt                                  # aB....txt
  ls *_{19,20}[0-9][0-9]-[01][0-9]-[0123][0-9]_*             # _YYYY-MM-DD_
  ls *-[012][0-9]:[0-5][0-9]:[0-5][0-9]-*                    # -HH:MM:SS-
```

Die sich bei der Expansion ergebende Kommandozeile ist allerdings in ihrer LÄNGE beschränkt (z.B. 2 Mio Zeichen bei der bash). Überschreitet man diese erlaubte Maximallänge, so wird das Kommando nicht ausgeführt und man erhält statt dessen eine Fehlermeldung der Form:

```
  argument list too long   # oder
  command line too long    #
```

Die maximale Kommandozeilenlänge erhält man per "xargs --show-limits":

```
  Your environment variables take up 1733 bytes
  POSIX upper limit on argument length (this system): 2093371
  POSIX smallest allowable upper limit on argument length (all systems): 4096
  Maximum length of command we could actually use: 2091638
  Size of command buffer we are actually using: 131072
  Maximum parallelism (--max-procs must be no greater): 2147483647
```

Soll KEIN Globbing eines Suchmusters erfolgen (weil z.B. ein Dateiname Metazeichen enthält), so ist es durch QUOTIERUNG aller oder einzelner Metazeichen mit "...", '...' oder "\" zu schützen.

```
  ls "*?[]"    # Dateiname *?[]
  ls '*?[]'    # Dateiname *=[]
  ls \*\?\[\]  # Dateiname *=[]
```

Mehrere Muster sind durch WHITESPACE (LEERRAUM = Leerzeichen, Tabulator, Newline) zu trennen. Sie werden von der Shell einzeln für sich expandiert und die Ergebnisse in die endgültige Kommandozeile eingesetzt.

```
  ls *.sh *.awk *.pl tmp*
  ls    *.sh    *.awk    *.pl    tmp*
```

VERSTECKTE Dateinamen mit führendem Punkt "." werden nur dann gefunden, wenn der Punkt explizit angegeben wird. D.h. die Muster "*" und "?" matchen keinen führenden Punkt "." in einem Dateinamen.

```
  ls .*       # Alle Punkt-Dateinamen INKLUSIVE "." und ".."
  ls .[^.]*   # Alle Punkt-Dateinamen AUSSER "." und ".."
```

ACHTUNG: Die Verzeichnisse "." (aktuelles Verzeichnis und ".." (Elternverzeichnis) beginnen ebenfalls mit einem Punkt!

```
  ls -d .*    # Alle Datei- und Verz.namen mit führendem Punkt
```

Auch Muster für ganze Dateipfade (Verzeichnis + Dateiname) sind angebbar, sie werden von der Shell bis zum (bitteren) Ende expandiert:

```
  ls *.sh                                  # Shell-Skripte im aktuellen Verz.
  ls /*/*.sh                               # im Unterverz. des akt. Verz.
```

```
  ls /*/*/*.sh                              # im Unter-unterverz. des akt. Verz.
  ls /*/*/*/*.sh                            # im Unter-unter-unterverz. ...
  ls *.sh /*/*.sh /*/*/*.sh /*/*/*/*.sh     # Alle von oben zusammen
  ls **.sh                                  # analog (Verz. beliebig tief)
  ls /usr/local/{bin,sbin}/*.sh             #
```

Solange die Länge der generierten Liste von Dateinamen unter der maximal möglichen der aktuellen Shell liegt (etwa 2 Mio Zeichen), werden alle passenden Dateinamen dafür von der Shell eingesetzt.

3) Tipps

* Die Option "-d" (directory) verhindert bei "ls" das Auflisten des INHALTS von Verzeichnissen, nur das Verzeichnis selbst wird ausgegeben (z.B. ls -ld /bin --> Verzeichnis "X11" in "/bin").

* Statt "ls" ist auch "echo" zum Auflisten der zu einem Suchmuster passenden Dateinamen nutzbar (Bsp: echo *.sh), da die Shell das Suchmuster expandiert.

* Die Verzeichnisse "/usr/bin" und "/usr/sbin" eignen sich aufgrund der vielen Dateien darin sehr gut zum Ausprobieren der Suchmuster.

4) Beispiele

HINWEIS: Immer "ls -d" oder "echo" voransetzen:

Muster	Dateinamen ...
a*	... mit "a" am Anfang (auch nur "a")
*a	... mit "a" am Ende (auch nur "a")
a	... mit mind. 1 "a" (auch nur "a")
a*a	... mit mind. 2 "a" am Anfang und am Ende (auch nur "aa")
*a*a*	... mit mind. 2 "a" (auch nur "aa")
aa	... mit mind. 2 "a" direkt hintereinander (auch nur "aa")
[a-z][a-z]	... der Länge 2 klein geschrieben
[0-9][0-9]*	... mit mind. 2 Ziffern
*a*e*i*o*u*	... mit mind. 5 Vokalen in angegebener Reihenfolge
?	... der Länge 1
??	... der Länge 2
?a?	... der Länge 3 und "a" als 2. Buchstaben
a?a	... der Länge 3 und "a" als 1. und 3. Buchstaben
? ??	
*.c	... mit Endung ".c"
/*/*.c	... mit Endung ".c" in Unterverz. des Root-Verz.
/*/*/*.c	... mit Endung ".c" in Unter-Unterverz. des Root-Verz.
/**.c	... mit Endung ".c" in ALLEN Unterverz. des Root-Verz.
*.[ch]	... die auf ".c" oder ".h" enden
*[!.][!ch]	... die als vorletztes Zeichen nicht "." und als letztes nicht "c" oder "h" haben (z.B. "a.", ".b", "aa")
*.{c,h,sh}	... die auf ".c", ".h" oder ".sh" enden
*.[ch] *.sh	... (analog)
.[!.]*	... mit führendem Punkt (versteckte, nicht "." und "..")
.*	... mit führendem Punkt (versteckte, inkl. "." und "..")