HOWTO zum Kommando "xargs"

$Id: xargs-HOWTO.txt,v 1.10 2025/02/23 20:14:55 tsbirn Exp $

Dieses Dokument beschreibt das Kommando "xargs", mit dessen Hilfe Kommandos, die die maximale Kommandozeilenlänge überschreiten würden, in mehrere Kommandos aufgeteilt werden, die sie nicht überschreiten (möglichst optimal, um möglichst wenig Prozesse zu starten).

Zu lange Kommandozeilen können sich beim Einsatz von Variablen-Substitution ($VAR), von Kommando-Substitution (`...` $(...)) und bei der Expansion von Shell-Mustern (* ? [...]) ergeben.

INHALTSVERZEICHNIS

---

1) Beschreibung

Das Kommando "xargs" ist ein sogenanntes "Helper-Tool", das ein beliebiges (anderes) Kommando CMD auf eine Liste von Dateien/Verzeichnissen anwendet, die ihm durch bestimmte Zeichen getrennt auf der Standard-Eingabe übergeben werden:

```
CMD... | xargs [OPT...] CMD [ARGS...]
xargs [OPT...] CMD [ARGS...] < DATEI
```

Dadurch soll für große Mengen von Dateien/Verzeichnissen die Anzahl der Kommando-Aufrufe bzw. der erzeugten Prozesse minimiert und gleichzeitig die maximale Länge der Kommandozeile (etwa 1-2 Mio Zeichen) ausgenutzt, aber nicht überschritten werden.

"xargs" bricht die Ausführung ab, falls keine Kommandozeile erstellt werden kann, CMD nicht aufrufbar ist, CMD von einem Signal beendet wird oder CMD mit Exit-Status 255 endet.

Ergibt Exit-Status 0 bei Erfolg, 127 falls CMD nicht gefunden wird und 126 falls CMD nicht ausführbar ist. Alle anderen Fehler ergeben Exit-Status 1.

Standard für CMD ist "echo".

HINWEIS: CMD darf nicht von der Standard-Eingabe lesen.

2) Optionen

Option	Name	Bedeutung
-0	--null	Trennzeichen NUL-Byte statt Leerzeichen/Newlines (passend zu -print0 in "find")
-E EOF	endof	Text EOF ist logisches Ende der Liste
-I STR	insert	Platzhalter STR für Dateiname festlegen (z.B. "%")
-J STR	join	Platzhalter STR für Zeilenliste festlegen (z.B. "%")

```
  | -L ANZ  | lines   | ANZ Zeilen pro CMD-Aufruf zusammenfassen (kein -n)   |
  | -n ANZ  | number  | ANZ Argumente pro CMD-Aufruf zusammenfassen (kein -L)|
  | -o      | opentty | Std-Input als "/dev/tty" in Kind-Prozessen öffnen    |
  | -p      | prompt  | CMD nur ausführen, falls "y" eingegeben wird (-t)    |
  | -r      | run     | CMD nur starten, falls mind. ein Parameter vorhanden |
  | -P MAX  | process | Max. MAX Instanzen von CMD parallel starten          |
  | -R REPL | replace | Max. Anz. einzusetzender Argumente bei -I            |
  | -s N    | size    | Max. Anz. Bytes für CMD-Zeile (STD: ARG_MAX = 4096)  |
  | -t      | type    | Kommando vor Ausführung ausgeben                     |
  | -x      | exit    | Abbruch falls erzeugtes Kommando zu lang ist (-s)    |
  +---------+---------+------------------------------------------------------+
  | --show-limits     | Grenzen von "xargs" (Kmdozeilenlänge, ...) ausgeben  |
  | --interactive     | Analog -p                                            |
  +-------------------+------------------------------------------------------+
```

Typische Ausgabe von "xargs --show-limits":

```
  Your environment variables take up 1786 bytes
  POSIX upper limit on argument length (this system): 2093318
  POSIX smallest allowable upper limit on argument length (all systems): 4096
  Maximum length of command we could actually use: 2091532
  Size of command buffer we are actually using: 131072
  Maximum parallelism (--max-procs must be no greater): 2147483647
```

Auf Deutsch:

```
  Die Umgebungsvariablen beanspruchen 2171 Bytes.
  In POSIX erlaubte Obergrenze der Argumentlänge (für dieses System): 2092933
  In POSIX erlaubte Obergrenze für die Argumentlänge (alle Systeme): 4096
  Maximale tatsächlich nutzbare Befehlslänge: 2090762
  Größe des tatsächlich verwendeten Befehlspuffers: 131072
  Maximaler Parallelismus (--max-procs darf nicht größer sein): 2147483647
```

3) Einsatz von "xargs"

Beim Matchen von Dateinamen per Dateinamen-Expansion kann leicht eine zu lange Kommandozeile entstehen (z.B. mehr als 2 Mio Zeichen):

```
  ls -l /*.c /*/*.c /*/*/*.c /*/*/*/*.c /*/*/*/*/*.c   # --> "argument list too long"
  #      1    2      3        4          5              # --> "command line too long"
```

Ersatzlösung mit "find":

```
  find / -maxdepth 5 -name "*.c" -ls
  find / -maxdepth 5 -name "*.c" -exec ls -l {} \;
```

"find" startet das nach "-exec/-ok" angegebene Kommando für JEDE gefundene Datei neu. Dies ist bei sehr vielen gefundenen Dateien ineffektiv, da jedesmal ein neuer Prozess erzeugt wird (wie in folgendem Beispiel):

```
  find $HOME -type f -size +200 -exec gzip {} \;
```

Effektiver sind die folgenden Aufrufe (einfügen des Ergebnisses von "find" auf der Kommandozeile per Kommando-Substitution `...` oder $(...) bzw. sammeln der MAXIMAL möglichen Menge an Argumenten vor einem Aufruf von "gzip" per "xargs"):

```
  gzip `find $HOME -type f -size +200 -print`          # 1) sh
  gzip $(find $HOME -type f -size +200 -print)         # 2) bash, ksh
  find $HOME -type f -size +200 -print | xargs gzip    # 3) xargs
```

Allerdings kann bei den Varianten 1)+2) ein "Überlaufproblem" eintreten, falls die einzufügende Liste zu lang ist (abhängig von der Shell ab 1-2 Mio Zeichen oder mehreren 1000 Namen).

4) Trennung per NUL-Byte "\0"

Alle obigen Varianten 1)-3) haben weiterhin den Nachteil, dass bei bestimmten Sonderzeichen in Dateinamen (Whitespace, ...) die erzeugte Liste falsch interpretiert wird. Bei GNU-"find" und GNU-"xargs" gibt es dafür eine Lösung, die als Trennzeichen das NUL-Byte "\0" verwendet (statt Zeilenvorschub "\n"), welches PRINZIPIELL nicht in einem Dateinamen vorkommen kann (neben dem "/"):

```
  find $HOME -type f -size +200 -print0 | xargs -0 gzip       # oder
  find $HOME -type f -size +200 -print0 | xargs --null gzip   #
```

HINWEIS: Viele weitere Kommandos, die Listen von Dateinamen verarbeiten, kennen inzwischen diesen Schalter -0/--null (z.B. tar, cpio, grep, sed, ...).