Bash String Manipulation

By Paul Kerling and Nikodem Staniucha

1. Bash String Split

In Bash kannst du einen String in ein Array aufteilen, indem du die Variable IFS (Internal Field Separator) und den Befehl read verwendest. Hier ist ein Beispiel:

```
string="Hallo Welt"
IFS=" " read -ra array <<< "$string"

# Gib jedes Element des Arrays aus
for element in "${array[@]}"
do
    echo "$element"
done
```

In diesem Beispiel setzen wir die Variable IFS auf ein Leerzeichen, um anzuzeigen, dass wir den String an jeder Leerstelle aufteilen möchten. Dann verwenden wir den Befehl read mit der Option -a, um den Eingabestring zu lesen und die aufgeteilten Teile in einem Array namens array zu speichern. Schließlich durchlaufen wir die Elemente des Arrays in einer Schleife und geben sie aus.
Du kannst den Wert von IFS anpassen, um den String mit verschiedenen Trennzeichen, wie Kommas oder Tabs, aufzuteilen. Mit <<< geben wir den Inhalt von string an read weiter, so dass read den String verarbeiten kann.
read liest den übergebenen String und teilt ihn an den Trennzeichen, die durch IFS festgelegt sind, auf. Die Optionen -r und -a haben folgende Bedeutungen:

- -r: Verhindert, dass read den Backslash () als Escape-Zeichen interpretiert. Das bedeutet, dass Backslashes in der Eingabe als normale Zeichen behandelt werden.
- -a array: Weist den aufgeteilten Teilen der Variable array zu. Dabei wird ein Array erstellt, das die einzelnen Teile des aufgeteilten Strings enthält.

2. Bash String Comparison

In Bash kannst du Zeichenfolgen mithilfe verschiedener Vergleichsoperatoren vergleichen. Hier sind einige gängige Methoden zum Vergleichen von Zeichenfolgen in Bash:

- Gleichheit (==): Du kannst das doppelte Gleichheitszeichen (==) verwenden, um zu überprüfen, ob zwei Zeichenketten gleich sind:

```
if [ "$string1" == "$string2" ]; then
    echo "Die Zeichenketten sind gleich"
fi
```

- Ungleichheit (!=): Der "Ungleich"-Operator (!=) überprüft, ob zwei Zeichenketten nicht gleich sind:

```
if [ "$string1" != "$string2" ]; then
    echo "Die Zeichenketten sind nicht gleich"
fi
```

- Größer als (>), kleiner als (<), größer oder gleich (>=), kleiner oder gleich (<=): Diese Operatoren vergleichen die Zeichenketten lexikographisch, basierend auf ihrer alphabetischen Reihenfolge. Zum Beispiel:

```
if [[ "$string1" > "$string2" ]]; then
    echo "Zeichenkette 1 ist größer als Zeichenkette 2"
fi
```

- Längenvergleich: Du kannst die Längen von zwei Zeichenketten vergleichen, indem du die -z (leer) und -n (nicht leer) Optionen des test-Kommandos verwendest:

```
if [ -z "$string" ]; then
    echo "Die Zeichenkette ist leer"
fi

if [ -n "$string" ]; then
    echo "Die Zeichenkette ist nicht leer"
fi
```

- Musterabgleich: Bash unterstützt Musterabgleiche mit dem == Operator und der [[…]] Konstruktion. Du kannst Platzhalterzeichen (* und ?) verwenden, um Muster abzugleichen. Zum Beispiel:

```
if [[ "$string" == "Hallo"* ]]; then
    echo "Die Zeichenkette beginnt mit 'Hallo'"
fi
```

3. *Bash Multi-Line Strings*

In Bash kannst du Mehrzeilige Strings auf verschiedene Arten erstellen. Hier sind einige Methoden:

- Mit einem einfachen String und Zeilenumbrüchen in einfachen Anführungszeichen:

```
string='Das ist ein mehrzeiliger
String in Bash.
Du kannst einfache Anführungszeichen
verwenden, um Zeilenumbrüche beizubehalten.'
```

- Mit einem doppelten String und escaped (maskierten) Zeilenumbrüchen:

```
string="Das ist ein mehrzeiliger \
String in Bash. \
Du kannst doppelte Anführungszeichen \
mit escaped Zeilenumbrüchen verwenden."
```

- Mit einem sogenannten "Here Document":

```
string=$(cat <<EOF
Das ist ein mehrzeiliger
String in Bash.
Du kannst ein Here Document
verwenden, um mehrzeilige Strings zu definieren.
EOF
)
```

EOF: Dies ist das Endemarkierungsschlüsselwort des "Here Document". Es gibt an, dass das "Here Document" abgeschlossen ist und der Inhalt bis hierhin in die Variable "string" geschrieben wird.
Beachte, dass das Schlüsselwort EOF, das im "Here Document" verwendet wird, beliebig gewählt werden kann, solange es auf beiden Zeilen gleich ist.

4. ***Bash String Replacement***

In Bash kannst du eine Zeichenkette mit dem sed-Befehl ersetzen. Hier ist die grundlegende Syntax:

```
sed 's/suchmuster/ersatzmuster/' eingabedatei
```

Dieser Befehl ersetzt das erste Vorkommen des suchmuster durch das ersatzmuster in der angegebenen eingabedatei. Wenn du die Ersetzung direkt in der Datei vornehmen möchtest (also die Datei direkt ändern möchtest), kannst du die -i-Option verwenden:

```
sed -i 's/suchmuster/ersatzmuster/' eingabedatei
```

Wenn du alle Vorkommen des suchmuster in der Datei ersetzen möchtest, kannst du die g-Flagge am Ende des sed-Befehls verwenden:

```
sed 's/suchmuster/ersatzmuster/g' eingabedatei
```

Hier ist ein Beispiel zur Veranschaulichung. Angenommen, du hast eine Datei namens beispiel.txt mit folgendem Inhalt:

```
Hallo, Welt! Dies ist eine Beispiels Zeichenkette.
```

Wenn du das Wort "Welt" in dieser Datei durch "Universum" ersetzen möchtest, kannst du den folgenden Befehl ausführen:

```
sed -i 's/Welt/Universum/' beispiel.txt
```

Nach Ausführung dieses Befehls wird der Inhalt von beispiel.txt wie folgt aussehen:

```
Hallo, Universum! Dies ist eine Beispiels Zeichenkette.
```

5. ***Bash Substring Extraction***

Hier sind die Beispiele zum Extrahieren von Teilstrings in Bash auf Deutsch:

- Mit Parameter-Erweiterung:

```
string="Hallo, Welt!"
teilstring=${string:7:5}
echo $teilstring
```

Ausgabe: Welt!

n diesem Beispiel bedeutet ${string:7:5}, dass ein Teilstring ab dem 7. Index (inklusiv) mit einer Länge von 5 Zeichen extrahiert wird.

- Mit dem expr-Befehl:

```
string="Hallo, Welt!"
teilstring=$(expr substr "$string" 8 5)
echo $teilstring
```

Ausgabe: Welt

Der expr-Befehl ermöglicht es, einen Teilstring mit der Funktion substr zu extrahieren. In diesem Fall wird ab der Position 8 gestartet und es werden 5 Zeichen extrahiert.

- Mit dem cut-Befehl:

```
string="Hallo, Welt!"
teilstring=$(echo "$string" | cut -c 8-12)
echo $teilstring
```

Ausgabe: Welt

Der cut-Befehl kann ebenfalls verwendet werden, um einen Teilstring zu extrahieren. -c 8-12 gibt den Bereich der Zeichen an, die extrahiert werden sollen, beginnend ab der 8. Position bis zur 12. Position.