3.2 Lektion 2

Zertifikat:	Linux Essentials
Version:	1.6
Thema:	3 Die Macht der Befehlszeile
Lernziel:	3.2 Daten in Dateien suchen und extrahieren
Lektion:	2 von 2

Einführung

In dieser Lektion werden wir uns Werkzeuge zur Textmanipulation ansehen, die häufig von Systemadministratoren oder Programmen verwendet werden, um wiederkehrende Informationen automatisch zu überwachen oder zu identifizieren.

Suche innerhalb von Dateien mit `grep`

Das erste Werkzeug, das wir in dieser Lektion besprechen werden, ist der Befehl `grep`. `grep` ist die Abkürzung für "`global regular expression print`" und seine Hauptfunktion ist die Suche in Dateien nach einem angegebenen Muster. Der Befehl gibt die Zeile mit dem angegebenen Muster rot markiert aus.

```
$ grep bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
user:x:1001:1001:User,,,,:/home/user:/bin/bash
```

`grep` lässt sich, wie die meisten Befehle, über Optionen steuern. Hier sind die häufigsten:

`-i:` Die Suche ist case-insensitiv, ignoriert also Groß-/Kleinschreibung.

`-r:` Die Suche ist rekursiv (sie sucht in allen Dateien innerhalb des angegebenen Verzeichnisses und seiner Unterverzeichnisse).

`-c:` Die Suche zählt die Anzahl der Treffer.

`-v:` Kehrt die Übereinstimmung um und gibt Zeilen aus, die *nicht* dem Suchbegriff entsprechen.

`-E:` Schaltet erweiterte reguläre Ausdrücke ein (benötigt von einigen fortgeschritteneren Metazeichen wie `|`, `+` und `?`).

`grep` hat viele andere nützliche Optionen. Schauen Sie in die Man Page, um mehr darüber zu erfahren.

Reguläre Ausdrücke

Das zweite Werkzeug ist sehr leistungsfähig und dient der Beschreibung von Textteilen in Dateien, auch ***reguläre Ausdrücke*** genannt. Reguläre Ausdrücke sind äußerst **nützlich bei der Extraktion von Daten aus Textdateien** durch die Angabe von Mustern. Sie werden häufig in Skripten oder bei der Programmierung mit Hochsprachen wie Perl oder Python verwendet.

Bei der Arbeit mit regulären Ausdrücken ist es sehr wichtig zu beachten, **dass *jedes Zeichen zählt*** und dass das **Muster mit dem Ziel geschrieben wird**, eine bestimmte Zeichenfolge abzubilden, die als Zeichenkette oder *String* bezeichnet wird. Die meisten Muster verwenden die **normalen ASCII-Zeichen, wie Buchstaben, Ziffern, Satzzeichen oder andere Symbole,** aber sie **können auch Unicode-Zeichen** verwenden, um jede andere Art von Text abzubilden.

Die folgende Liste beschreibt die Metazeichen der regulären Ausdrücke, die zur Bildung der Muster verwendet werden:

`.` Übereinstimmung eines einzelnen Zeichens (außer Zeilenumbruch).

`[abcABC]` Übereinstimmung eines beliebigen Zeichens innerhalb der Klammern.

`[^abcABC]` Übereinstimmung eines beliebigen Zeichens außer denen innerhalb der Klammern.

`[a-z]` Übereinstimmung eines beliebigen Zeichens im angegebenen Bereich.

`[^a-z]` Übereinstimmung eines beliebigen Zeichens außer denen im angegebenen Bereich.

`sun|moon` Übereinstimmung mit einer der angegebenen Zeichenketten.

`^` Zeilenbeginn

`$` Zeilenende

Alle Funktionalitäten der regulären Ausdrücke lassen sich auch über `grep` implementieren. Im obigen Beispiel sehen Sie, dass das Wort nicht von doppelten Anführungszeichen umgeben ist. Um zu **verhindern, dass die Shell das Metazeichen selbst interpretiert**, wird empfohlen, **komplexere Muster in doppelte Anführungszeichen** (" ") zu setzen. Zu Übungszwecken verwenden wir bei der Implementierung regulärer Ausdrücke doppelte Anführungszeichen. Die anderen Anführungszeichen behalten ihre normale Funktionalität, wie in früheren Lektionen erläutert.

Die folgenden Beispiele verdeutlichen die Funktionalität der regulären Ausdrücke. Wir benötigen Daten in der Datei, daher hängt der nächste Befehlssatz einfach verschiedene Zeichenketten an die Datei `text.txt` an.

```
$ echo "aaabbb1" > text.txt
$ echo "abab2" >> text.txt
$ echo "noone2" >> text.txt
$ echo "class1" >> text.txt
$ echo "alien2" >> text.txt
$ cat text.txt
aaabbb1
abab2
noone2
class1
alien2
```

Das erste Beispiel ist eine Kombination aus der Suche in der Datei ohne und mit regulären Ausdrücken. Um reguläre Ausdrücke vollständig zu verstehen, ist es sehr wichtig, den Unterschied zu zeigen:

Der **erste Befehl** sucht nach der **genauen Zeichenkette** an beliebiger Stelle in der Zeile, während der **zweite Befehl** nach Zeichenketten sucht, die **eines der Zeichen** zwischen den Klammern enthalten. Darum liefern die Befehle unterschiedliche Ergebnisse.

```
$ grep "ab" text.txt
aaabbb1
abab2
$ grep "[ab]" text.txt
aaabbb1
abab2
class1
alien2
```

Die nächsten Beispiele zeigen die Anwendung der Metazeichen für Zeilenanfang und Zeilenende. Es ist sehr wichtig, die beiden Zeichen an die richtige Stelle im Ausdruck zu setzen: Bei der Angabe des **Zeilenanfangs muss das Metazeichen *vor* dem Ausdruck stehen**, während es bei der Angabe des **Zeilenendes *nach* dem Ausdruck stehen muss.**

```
$ grep "^a" text.txt
aaabbb1
abab2
alien2
$ grep "2$" text.txt
abab2
noone2
alien2
```

Neben den zuvor erläuterten Metazeichen haben reguläre Ausdrücke weitere Metazeichen zur Wiederholung eines angegebenen Musters:

`*` **Null oder mehr** Vorkommen des vorhergehenden Musters.

`+` **Ein oder mehr** Vorkommen des vorhergehenden Musters.

`?` **Null oder ein Vorkommen** des vorhergehenden Musters.

Als Beispiel für die Multiplikator-Metazeichen sucht der folgende Befehl nach einer Zeichenkette, die `ab`, ein einzelnes Zeichen und ein oder mehrere der zuvor gefundenen Zeichen enthält. Das Ergebnis zeigt, dass `grep` die Zeichenkette `aaabbb1` gefunden hat, die dem Teil `abbb entspricht, wie auch `abab2`. Da `+` ein Zeichen eines ***erweiterten* regulären Ausdrucks** ist, müssen wir dem Befehl `grep` die **Option `-E` mitgeben**.

```
$ grep -E "ab.+" text.txt
aaabbb1
abab2
```

Die meisten Metazeichen sind selbsterklärend, können aber bei der ersten Verwendung knifflig werden. Die vorherigen Beispiele stellen einen kleinen Teil der Funktionalität der regulären Ausdrücke dar. Probieren Sie alle genannten Metazeichen aus, um mehr darüber zu erfahren, wie sie funktionieren.

Geführte Übungen

Suchen Sie mit `grep` in der Datei `/usr/share/hunspell/en_US.dic` die Zeilen, die den folgenden Kriterien entsprechen:

1. Alle Zeilen, die die Zeichenfolge `cat` an beliebiger Stelle in der Zeile enthalten.
2. Alle Zeilen, die keines der folgenden Zeichen enthalten: `sawgtfixk`.
3. Alle Zeilen, die mit 3 beliebigen Buchstaben und der Zeichenfolge `dig` beginnen.
4. Alle Zeilen, die mit mindestens einem `e` enden.
5. Alle Zeilen, die eine der folgenden Zeichenfolgen enthalten: `org`, `kay` oder `tuna`.
6. Anzahl der Zeilen, die mit einem oder keinem `c` beginnen, gefolgt von der Zeichenkette `ati`.

Offene Übungen

1. Finden Sie den regulären Ausdruck, der mit den Wörtern in der Zeile “Einschließen” übereinstimmt und nicht mit denen in der Zeile “Ausschließen”:
    - Einschließen: `pot`, `spot`, `apot`
      Ausschließen: `potic`, `spots`, `potatoe`
    - Einschließen: `arp99`, `apple`, `zipper`
      Ausschließen: `zoo`, `arive`, `attack`
    - Einschließen: `arcane`, `capper`, `zoology`
      Ausschließen: `air`, `coper`, `zoloc`
    - Einschließen: `0th/pt`, `3th/tc`, `9th/pt`
      Ausschließen: `0/nm`, `3/nm`, `9/nm`
    - Einschließen: `Hawaii`, `Dario`, `Ramiro`
      Ausschließen: `hawaii`, `Ian`, `Alice`
2. Welcher andere nützliche Befehl wird häufig verwendet, um innerhalb von Dateien zu suchen. Welche zusätzlichen Funktionalitäten hat er?
3. Nutzen Sie ein Beispiel aus der vorangegangenen Lektion und versuchen Sie, mit Hilfe von `grep` nach einem bestimmten Muster in der Ausgabe des Befehls zu suchen.

Zusammenfassung

In dieser Lektion haben Sie gelernt:

- Reguläre Ausdrücke und Metazeichen.
- Wie man Muster mit regulären Ausdrücken erstellt.
- Wie man in Dateien sucht.

Befehle, die in den Übungen verwendet werden:

`grep`
: Sucht nach Zeichen oder Zeichenketten in einer Datei.