

Apropos

Mit *apropos* nach mehreren Begriffen suchen

Der *apropos* Befehl hat den kleinen Nachteil, dass er oft sehr viele Treffer liefert und zudem nicht ohne weiteres erlaubt, mehr als einen Suchbegriff anzugeben.

Um das Problem zu beheben, wollen wir seine Ausgabe durch den Filter `grep` schicken, um eine Art AND-Verknüpfung der Suchbegriffe zu erreichen.

Schreibt dazu ein Skript `myapropos`, das zwei Suchbegriffe als Argument nimmt und somit eine kürzere Liste als *apropos* zurückgibt.

Aufrufbeispiel:

```
herbert@Paulina:/home/herbert # ./myapropos find file
diff (1)                - find differences between two files
find (1)                 - search for files in a directory hierarchy
File::Find (3pm)        - traverse a file tree
XtFindFile (3x)         - search for a file using substitutions in the path list
diff (1)                - find differences between two files
find (1)                 - search for files in a directory hierarchy
File::Find (3pm)        - traverse a file tree
```

Dass hier möglicherweise einige Einträge mehrfach vorkommen, liegt daran, dass sie auf manchen Maschinen mehrfach installiert sind, das ist also nichts Beunruhigendes, Hauptsache, es sind alle angezeigten Einträge mindestens einmal da.

Auf Fehlerbehandlung wird verzichtet, es wird also nicht weiter überprüft, ob auch wirklich zwei Argumente übergeben wurden.

Das Resultat dürfte nicht länger als zwei Zeilen sein, macht Euch das Leben also nicht unnötig schwer.

Suchen in Konfigurationsdateien

Unter Unix befinden sich die meisten systemweiten Konfigurationsdateien unter `/etc`. Irgendwo dort wird auch der Webserver `apache` konfiguriert.

Schreibt ein Skript `pubfind`, das das Verzeichnis `/etc` und alle seine Unterverzeichnisse nach Dateien durchsucht, die mit `.conf` aufhören. Verwendet hierzu den `find`-Befehl.

Lest die Online-Dokumentation zu diesem Befehl und findet heraus, wie das Ding arbeitet.

Wichtig: Das Skript muss auch dann funktionieren, wenn es aus dem `/etc`-Verzeichnis heraus aufgerufen wird, das ist nicht trivial

(Hinweis: hier mit `cd`-Aufrufen im Skript zu schummeln, ist nicht erlaubt)!

Die Dateiliste, die `find` liefert, soll nach der Zeichenkette `public_html` durchsucht werden, so dass folgendes Ergebnis herauskommt:

```
herbert@Paulina:/home/herbert # ./pubfind
/etc/httpd/httpd.conf:      UserDir public_html
/etc/httpd/httpd.conf:<Directory /home/*/public_html>
```

Last Euch nicht davon durcheinanderbringen, dass es in `/etc` einige Dateien und Verzeichnisse gibt, auf die Normalsterbliche keinen Zugriff haben.

Unterdrückt daher die Fehlermeldungen bei der Ausgabe.

Auch hier sollte es Euch nachdenklich machen, wenn das Ergebnis länger als zwei Zeilen ist.

Hinweise

Skripte

Ein Skript ist eine Textdatei, die Shell-Kommandos enthält.

Am Anfang jedes Skriptes (in der physikalisch ersten Zeile), sollte immer der Name und Pfad des Interpreters festgelegt werden:

```
#!/bin/sh
```

Das Skript (z.B. `meinskript`) sollte dann ausführbar gemacht werden:

```
chmod +x meinskript
```

Es kann dann ausgeführt werden mit:

```
./meinskript
```

Die explizite Pfadangabe dient dazu, ganz sicherzustellen, dass auch wirklich das Programm aus dem aktuellen Verzeichnis geladen wird.

Auf Kommandozeilenargumente kann über die speziellen Variablen `$1`, `$2` usw. zugegriffen werden.

- UNIX-Übungen WS00/01-

Pipes und Backquotes

In dieser Übung werden zwei Kommunikationsmechanismen der Shell zum Einsatz kommen, die in der Vorlesung auch schon einige Male aufgetaucht sind: die Pipe (senkrechter Strich |) und die Kommandoersetzung (Backquotes `...`).

Welche wesentlichen Unterschiede gibt es zwischen diesen beiden Mechanismen? Wichtig zum Verständnis sind die Fragen, ob die verbundenen Prozesse gleichzeitig oder nacheinander aufgerufen werden und auf welche Art die Information weitergeleitet wird.

Tests bei der Abnahme

Beide Skripte werden bei der Abnahme automatisch getestet.

Hierzu werden wir das von Daniel und Gerrit entwickelte Tool `arnold` (Another Ridiculous Normalization Or Liquidation Device) zum Einsatz kommen. Ladet Euch das Ding herunter und probiert ein wenig damit herum.

Die Tests für `arnold` werden in Testdateien im Textformat, sogenannten Testbenches, definiert. Um warm zu werden, stellen wir Euch für diese Aufgabe genau die Testdateien zur Verfügung, mit denen wir später auch abnehmen werden, bei späteren Aufgaben werden wir dann mit leicht erweiterten Testbenches arbeiten, so dass Ihr selber ein wenig Kreativität im Ausdenken von Tests entwickeln müsst:

für Aufgabe 1 (`myapropos`)
für Aufgabe 2 (`pubfind`)

Der Aufruf sieht dann wie folgt aus (vorher `arnold` mit `chmod` ausführbar machen):

```
herbert@Paulina:/home/herbert # ./arnold ./myapropos myapropos.arnold

PARAM: 'find file'; EXP: 'diff'; OUT: 'diff (1)      - find differences between two files
find (1)                                           - search for files in a directory hierarchy
File::Find (3pm)                                   - traverse a file tree
XtFindFile (3x)                                    - search for a file using substitutions in the path list
iff (1)                                             - find differences between two files
find (1)                                           - search for files in a directory hierarchy
File::Find (3pm)                                   - traverse a file tree' ==> --- FindDiff (-case) OK ---
PARAM: 'find file'; EXP: 'File::Find'; OUT: 'diff (1) - find differences between two files
find (1)                                           - search for files in a directory hierarchy
File::Find (3pm)                                   - traverse a file tree
XtFindFile (3x)                                    - search for a file using substitutions in the path list
diff (1)                                           - find differences between two files
find (1)                                           - search for files in a directory hierarchy
File::Find (3pm)                                   - traverse a file tree' ==> --- FindDiff (-case) OK ---

Test-Statistik:
=====
Testname: FindDiff (-case) OK
Testname: FindDiff (-case) OK
Alles klar, gut und so. Von dem Test IHM sein Ende!
```

Tip: Schaut Euch die Testdateien an und probiert mit den Tests herum. Führt auf jeden Fall auch mal bewusst einen Fehler herbei, um zu sehen, dass das Ding auch wirklich Fehler findet!

- UNIX-Übungen WS00/01-

myapropos:

```
#!/bin/sh
apropos $1 | grep $2
```

pubfind:

```
#!/bin/sh
grep public_html `find /etc -name "*.conf" 2>/dev/null` 2>/dev/null
```