Prof. Dr. Uwe Schmidt

ptlwedel
PRIVATE BERUFSFACHSCHULE

2. Februar 2016

Aufgaben zur Klausur **Unix** im WS 2015/16 (107, 351, 505, 506, 606)

Zeit: 75 Minuten erlaubte Hilfsmittel: keine

Bitte tragen Sie Ihre Antworten und fertigen Lösungen ausschließlich an den gekennzeichneten Stellen in das Aufgabenblatt ein. Ist ihre Lösung wesentlich umfangreicher, so überprüfen Sie bitte nochmals Ihren Lösungsweg.

Nutzen Sie die Rückseiten der Klausur zur Entwicklung der Lösungen und übertragen die fertigen Lösungen in das Aufgabenblatt.

Sollten Unklarheiten oder Mehrdeutigkeiten bei der Aufgabenstellung auftreten, so notieren Sie bitte, wie Sie die Aufgabe interpretiert haben.

Viel Erfolg!

Diese Klausur besteht einschließlich dieses Deckblattes aus 5 Seiten.

Aufgabe 1:

Gegeben sei folgendes bash-Skript:

```
#!/bin/bash

echo '#!/bin/bash'

for i in $*
do
    echo "echo $i 1>&2"
    echo "cat >$i <<'Ende von $i'"
    cat $i
    echo "Ende von $i"
done
```

Verfeinern Sie dieses Skript so, dass es auch mit Dateinamen korrekt arbeitet, die Leerraum und andere Sonderzeichen enthalten.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Aufgabe 2:

Entwickeln Sie einen Filter mit `sed`, der in einem Text alle Kommentare der Form `//...`, d.h. alle Zeichen auf einer Zeile die mit zwei Schrägstrichen beginnen und bis zum Zeilenende gehen, in Kommentare nach C–Konvention, `/* ... */` transformiert, also in eine Zeichenreihe, die mit `/*` beginnt und mit `*/` endet.

. . . . . .

. . . . . .

Aufgabe 3:

Entwickeln Sie ein Kommando, dass das gesammte Dateisystem nach Dateien durchsucht, die mit `.jpg` oder `.png` enden. Die Liste der Dateinamen soll in einer Datei `bilder` im momentanen Arbeitsverzeichnis gespeichert werden. Mögliche Fehlermeldungen, die beim Suchen auftreten können, sollen nicht angezeigt werden. Wärend der Suche soll normal weitergearbeitet werden können.

. . . . . .

. . . . . .

Aufgabe 4:

Gegeben sei eine Datei mit Wörtern für ein Wörterbuch. Die Datei sei so organinsiert, dass auf jeder Zeile genau ein Wort steht. Alle Wörter bestehen dabei nur aus den 26 Kleinbuchstaben des ASCII Zeichensatzes. In den zu entwickelnden regulären Ausdrücken müssen also nur diese 26 Zeichen berücksichtingt werden. Man findet solche Wörterbüchser zum Beispiel auf manchen Unix–Systemen unter `/usr/share/dict/words`.

Geben sie in den folgenden Aufgaben einen regulären Ausdruck an, mit dem man die jeweils gesuchten Wörter mit dem `egrep` Kommando selektieren kann. `egrep` oder auch `grep -E` verarbeiten den vollen Umfang der Syntax für reguläre Ausdrücke, `grep` alleine akzeptiert auf manchen Systemen nur eine eingeschränkte Syntax. Beachten Sie bitte, dass `egrep` im Standardverhalten nur nach Teilzeichenreihen sucht und alle Zeilen mit einer gefundenen Teilzeichenreihe ausgibt.

Nutzen Sie nur die in der Vorlesung verwendete Syntax für reguläre Ausdrücke, und keine Erweiterungen, die dort nicht behandelt worden sind.

1. Gesucht sind alle Wörter, die als 3. Buchstaben ein `c` haben und mit `xyz` enden.

   .....

2. Gesucht sind alle Wörter, die mindestens drei mal das Zeichen `x` enthalten.

   .....

3. Gesucht sind alle Wörter mit vier bis sechs Zeichen

   .....

4. Gesucht sind alle Wörter, die genau zwei mal das Zeichen `a` enthalten.

   .....

5. Gesucht sind alle Wörter, die ein Doppel-`b` oder ein Doppel-`p` enthalten.

   .....

6. Gesucht sind alle Wörter, die mindestens ein `e` und ein `i` enthalten.

   .....

7. Gesucht sind alle Wörter, die genau ein `a` und ein `o` enthalten, wobei das `a` als erstes im Wort vorkommen soll.

   .....

   .....

8. Gesucht sind alle Wörter, die genau ein `e` und ein `i` enthalten, wobei die Reihenfolge nicht festgelegt ist.

   .....

   .....

Aufgabe 5:

Gegeben seien folgende sechs C–Quellcode–Dateien `Test.c, Mod1.c, Mod2.c, Mod1.h, Mod2.h` und `Types.h`. `Test.c` enthält das Hauptprogramm, dieses benutzt Routinen, die in `Mod1.h` deklariert sind und in `Mod1.c` implementiert sind. `Mod1.c` benutzt Routinen, die in `Mod2.h` deklariert sind und in `Mod2.c` implementiert sind. Alle `.c`–Dateien verwenden globale Datendefinitionen aus `Types.h`

Schreiben Sie einen `Makefile` zum Erzeugen eines Programms `Test` aus den oben beschriebenen Dateien. Beachten Sie dabei alle Abhängigkeiten zwischen den Dateien, entwickeln Sie den `Makefile` aber so, daß keine überflüssigen Aktionen gemacht werden.

Nutzen Sie keine im `make`–System vordefinierten oder eingebauten Regeln.