Physikalisch–Technische Lehranstalt Wedel
Prof. Dr. Uwe Schmidt

4. Februar 2004

Aufgaben zur Klausur **Unix** im WS 2003/04 (IA 351)

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.

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 7 Seiten

---

Aufgabe 1:

Geben Sie ein UNIX-Kommando an, das zählt in wieviel Zeilen in allen .html–Dateien im momentanen Verzeichnis und allen Unterverzeichnissen das Wort `TITLE` vorkommt.

. . . . . .

. . . . . .

Aufgabe 2:

In einer Unix Umgebung verwendet man häufig Skript-Sprachen, *shell*–Skripte, Tcl, Perl, ..., insbesondere für WWW–Anwendungen.

Nennen Sie einige Vorteile dieser Sprachen gegenüber C– oder Pascal–Programmierung.

.....

.....

.....

.....

Nennen Sie Nachteile diser Sprachen gegenüber C– oder Pascal–Programmierung.

.....

.....

.....

.....

In MS–Windows Umbegungen verwendet man wenig diese Skript-Sprachen und fast ausschließlich grafische Oberflächen. Welche Nachteile dieser ausschließlich menügesteuerten Programme kennen Sie?

.....

.....

.....

.....

Aufgabe 3:

Ein UNIX–Prozeß besitzt im Betriebssystem 3 Datensegment, das Textsegment, das Benutzerdatensegment und das Systemdatensegment.

Was wird bei der Ausführung eines Programms im Textsegment gespeichert?

1) .....

2) .....

Was wird bei der Ausführung eines Programms im Benutzerdatensegment gespeichert?

1) .....

2) .....

3) .....

4) .....

Was wird bei der Ausführung eines Programms im Systemdatensegment gespeichert?

1) .....

2) .....

3) .....

4) .....

Welche Segmente sind im Benutzermodus schreibgeschützt?

.....

Welche Segmente sind im privilegierten Systemmodus schreibgeschützt?

.....

Was passiert mit den drei Segmenten bei einem Aufruf von `fork`.

.....

.....

.....

.....

.....

Aufgabe 4:

Reguläre Ausdrücke sind ein gutes Werkzeug zur Beschreibung von Textmustern. Für den Aufbau von regulären Ausdrücken gibt es folgende Regeln. Erweiterungen wie sie z.B. in Perl vorkommen sind hier nicht zugelassen.

1. jedes Zeichen aus dem Alphabeth (z.B. dem ASCII Zeichensatz) ist ein regulärer Ausdruck.
2. Sonderzeichen, die zum Aufbau der Ausdrücke verwendet werden, z.B. `\ [ ] . * + ( ) | ?`, müssen mit einem `\` maskiert werden. `\.` steht also für das Zeichen `.`
3. `.` steht für ein beliebiges Zeichen
4. `[`$z_1 z_2 z_3$`]` steht für eine Menge von Zeichen z_1, z_2, z_3 , hier dürfen auch Intervalle angegeben werden: `[`z_1`-`z_n`]`.
5. `[^`$z_1 z_2 z_3$`]` steht für die Menge aller Zeichen außer z_1, z_2 und z_3.
6. Wenn r_1 und r_2 reguläre Ausdrücke sind, dann auch $r_1 r_2$ (Folge, Sequenz, r_1 gefolgt von r_2)
7. Wenn r_1 und r_2 reguläre Ausdrücke sind, dann auch r_1`|`r_2 (Alternativen, Auswahl, r_1 oder r_2)
8. r_1`*` steht für die 0,1,2,...-fache Wiederholung von r_1
9. r_1`+` steht für die 1,2,...-fache Wiederholung von r_1
10. r_1`?` steht für r_1 oder die leere Zeichenfolge.
11. `(`r_1`)` steht für r_1, Klammern stehen also zum Zusammenfassen von regulären Ausdrücken.

Beispiele:

`[a-zA-Z0-9]`
: steht für die Menge der Buchstaben und Ziffern.

`[^&]`
: steht für die Menge aller Zeichen außer `&`.

`[^0-9]`
: alles außer Ziffern.

`ab*`
: Eine Zeichenfolge, die mit `a` als 1. Zeichen gefolgt von beliebig vielen `b`'s.

`ab+`
: Eine Zeichenfolge, die mit `a` als 1. Zeichen gefolgt von beliebig vielen `b`'s aber mindestens einem `b`.

`(ab)*`
: Eine Zeichenfolge, die abwechselnd aus `a`'s und `b`'s besteht.

`abc|def`
: entweder `abcef` oder `abdef`.

`(abc)|(def)`
: entweder `abc` oder `def`.

`a?`
: ein `a` oder nichts.

Aufgabe:

Beschreibung von Adressen von WWW-Dokumenten.
Ein Adresse eines WWW-Dokuments besteht aus einer manchmal etwas kryptischen Zeichenfolge, z.B.
`//www.fh-wedel.de:80/~si/index.html?NAME=wert&FOO=bar#abc`

Diese Adressen enthalten als erstes einen Rechnernamen eingeleitet durch `//`. Der vollständige Rechnername ist eine Folge von Namen getrennt durch einen `.` . Der Rechnername enthält Buchstaben, Ziffern und `-` .

Beschreiben Sie den Aufbau eines Rechnernamens durch einen regulären Ausdruck:

. . . . . .

Nach dem Rechnernamen kann optional eine Portnummer folgen, die durch einen `:` eingeleitet wird.

Geben Sie einen regulären Ausdruck für eine optionale Portnummer an:

. . . . . .

Danach folgt ein Dokumentenname, dieser besteht aus einem Pfad ähnlich dem eines absoluten Pfadnames im Dateisystem, darf aber keinen `#`und keine `?` enthalten.

Beschreiben Sie diesen Aufbau durch einen regulären Ausdruck:

. . . . . .

Optional kann noch ein Parameterstring kommen, dieser beginnt mit einem `?` und besteht aus einer Folge von Parametern, die durch `&` getrennt sind. Der Parameterstring darf keinen `#`enthalten.

Ein regulärer Ausdruck für den optionalen Parameterstring:

. . . . . .

Als letzter Bestandteil darf eine Marke für eine Position in einem Dokument folgen. Diese wird durch ein `#` eingeleitet und darf beliebige Zeichen enthalten.

Ein regulärer Ausdruck für eine optionale Marke:

. . . . . .