ulm university universität
uulm

Institut für Angewandte Informationsverarbeitung

Dr. Andreas F. Borchert
N.N.

18. Oktober 2012
Blatt 1

# Systemnahe Software I (WS 2012/2013)

**Abgabe bis zum 25. Oktober 2012, 16:00 Uhr**

**Lernziele:**

- Erste Vertrautheit mit der Unix-Shell und die
- Fähigkeit, ein einfaches C-Programm mit einem Texteditor unter Unix zu erstellen, zu übersetzen und auszuführen.

## Hinweis:

Melden Sie sich für diese Veranstaltung in SLC an. Ohne diese Anmeldung können Sie keine Lösungen elektronisch einreichen und Sie können auch nicht von Ihren Kommilitonen als Team-Mitglied genannt werden. Sobald Sie sich angemeldet haben, wird es bis zu einer Stunde brauchen, bis Sie anschließend Lösungen einreichen können.

## Aufgabe 1: Erste Schritte mit der Shell und mit einem Texteditor (5 Punkte)

Arbeiten Sie auf einem der Arbeitsplätze in unseren Pools oder melden Sie von anderswo per SSH (*secure shell*) auf *thales.mathematik.uni-ulm.de* an. Sobald die Anmeldung erfolgreich war, sollten Sie den Prompt der Shell sehen. Per Voreinstellung ist dies bei uns der jeweilige Rechnername, gefolgt von einem Dollarzeichen, also beispielsweise „thales$ “.

Kommandos beginnen mit dem Namen des Kommandos und weiteren Parametern. Ein einfaches Kommando ist beispielsweise *date*, das das aktuelle Datum ausgibt:

```
thales$ date
Thu Oct 18 10:50:50 MEST 2012
thales$
```

Wenn Sie sich über diese oder andere Kommandos näher informieren möchten, helfen die Manualseiten. Die Manualseite zu *date* können Sie mit „man date“ abrufen. Probieren Sie das aus und ermitteln Sie mit Hilfe der Manualseite, wie das *date*-Kommando so aufgerufen

werden kann, dass *date* nur das Datum in der Form Tag, Monat, Jahr, getrennt mit Punkten ausgibt, also beispielsweise „18.10.2012“.

Hinweis: Wenn Sie die Manualseite lesen, befinden Sie sich in einem Programm zum seitenweisen Lesen von Text, das sich *less* nennt. Mit der Leertaste können Sie vorwärts blättern, mit „b“ rückwärts gehen, mit „h“ erhalten Sie einen Hilfetext und mit „q“ können Sie das Lesen beenden, so dass Sie wieder in die Shell zurückkehren.

Falls Sie noch nie einen Texteditor unter Unix benutzt haben, ist es jetzt an der Zeit, dies zu lernen. Empfehlenswert und besonders geeignet für Software-Entwicklung wäre insbesondere der *vim*. Die ersten Schritte lassen sich am besten mit dem *vitutor* erlernen. Sie können aber notfalls auch auf den *pico* zurückgreifen, der von der ersten Benutzung an vollkommen selbsterklärend ist und deswegen zu Beginn etwas einfacher erscheint, obwohl er auf lange Sicht deutlich hinter den Möglichkeiten des *vim* zurückbleibt.

Erstellen Sie mit dem Texteditor eine Datei namens *date.txt* und beschreiben Sie darin, wie *date* so aufgerufen werden kann, dass die Ausgabe in der oben beschriebenen Weise erfolgt. Diese Datei können Sie danach elektronisch einreichen:

```
thales$ submit ss1 1 date.txt
thales$
```

Wenn es geklappt hat, gibt es hier keine Fehlermeldung. Wenn es mit der Registrierung bei SLC noch nicht geklappt hat, erscheint die Meldung „submit: you are not yet registered for ss1“.

Sie können und sollen jedoch im Team arbeiten, wobei bis zu vier einem Team angehören können und drei bis vier Mitglieder die erwünschte Größe eines Teams ist. Wenn Sie eine Lösung elektronisch einreichen, sollten Sie darauf achten, dass diese auch ihren Team-Mitgliedern angerechnet wird. Dies erfolgt durch das Anlegen einer *team*-Datei. In dieser steht in jeder Zeile der Benutzername eines Team-Mitglieds. Ihren eigenen Benutzernamen müssen Sie dabei nicht angeben, es schadet aber auch nichts, wenn er dabei ist. Es können aber nur solche Team-Mitglieder angegeben werden, die auch für die Vorlesung registriert sind. Dann können Sie bei dem *submit*-Befehl auch die *team*-Datei mit angeben:

```
thales$ submit ss1 1 date.txt team
thales$
```

Wenn beispielsweise „alice“ in der *team*-Datei genannt wird und es keinen Benutzer mit Shell-Zugang namens „alice“ gibt, dann erscheint die Fehlermeldung „submit: alice found in team but this user is unknown“. Sollte „alice“ einen Zugang bei uns haben, aber noch nicht für die Vorlesung registriert sein, erscheint die Meldung „submit: alice found in team but this user is not registered“.

Testen Sie es also bitte erneut mitsamt einer *team*-Datei. Beachten Sie dabei, dass Sie beliebig oft eine Lösung elektronisch einreichen können, aber nur die letzte Abgabe zählt.

### Aufgabe 2: Ein erstes C-Programm (5 Punkte)

Zu den ersten Versuchen in einer neuen Programmiersprache gehört typischerweise ein Programm, das einen mit einer Ausgabe begrüßt. Versuchen Sie also ein C-Programm mit dem

Dateinamen „hello.c“ zu erstellen, das eine beliebige Begrüßung ausgibt. Sie finden dazu Beispiele im Skript oder auch auf dem Netz, wenn Sie nach „hello world“ suchen. Sie können Ihr C-Programm mit dem Kommando *gcc* übersetzen:

```
thales$ gcc -Wall -std=gnu99 -o hello hello.c
```

Die Option „-Wall“ schaltet alle Warnungen ein, mit „-std=gnu99“ wird C99 (das ist der letzte C-Standard) zugrundegelegt mitsamt einigen GNU-Erweiterungen, mit „-o hello“ wird der Name des zu erzeugenden ausführbaren Programms bestimmt. Wenn es klappt, gibt es keine Meldung (frei nach der Philosophie *no news are good news*) und die Datei *hello* ist erzeugt. Im Erfolgsfalle können Sie das erzeugte Programm ausführen, indem Sie *hello* aufrufen:

```
thales$ hello
Hello world!
thales$
```

Mit dem Werkzeug *truss* unter Solaris und *strace* unter Linux können Sie ein Protokoll aller Systemaufrufe eines Programms erzeugen. Hiermit wird Ihr Programm unter der Kontrolle von *truss* aufgerufen und in der Datei *syscalls1.log* das Protokoll abgelegt:

```
thales$ truss hello 2>syscalls1.log
```

(Mit „2>“ wird die Standardfehlerausgabe in eine Datei umgelenkt.) Sie können sich die Datei mal neugierigerweise ansehen und nach dem Systemaufruf *write* suchen, mit dem die Begrüßung ausgegeben wurde. Zum Vergleich können Sie ein ähnliches Begrüßungsprogramm in anderen Sprachen schreiben, etwa in Java und dies zum Vergleich testen:

```
thales$ javac hello.java
thales$ truss java hello 2>syscalls2.log
Hello world!
thales$
```

Wieviele Systemaufrufe benötigt C und wieviele Java? Wieviel Aufrufe von *write* sind jeweils enthalten? Schreiben Sie Ihre Erkenntnisse in eine Datei namens *syscalls.txt*.

Wenn alles geklappt hat, können Sie mit *rm hello* das ausführbare Programm wieder löschen (*rm* steht für *remove*) und danach mit *submit* Ihre Lösung einreichen:

```
thales$ submit ss1 2 hello.c syscalls.txt team
```

## Viel Erfolg!