Objektorientierte Programmierung mit C++ (WS 2010)

Dr. Andreas F. Borchert, Tobias Brosch
Institut für Angewandte Informationsverarbeitung
Universität Ulm

Blatt 1: Abgabetermin 27. Oktober 2010

1 Shell

Zu Beginn lohnt es sich auf eine ausgewachsene IDE (Integrated Development Environment) zu verzichten. Zum Zweck der Übungen empfehlen wir sich auf den Server

```
theseus.mathematik.uni-ulm.de
```

zu begeben und auf der Shell zu arbeiten. Unter Linux/Unix Systemen (z.B. die Rechner im Pool) geht dies durch Aufruf des Programms `ssh` in einer Shell mit dem Argument `login@theseus.mathematik.uni-ulm.de`:

```
ssh login@theseus.mathematik.uni-ulm.de
```

wobei `login` durch den eigenen Login-Namen ersetzt werden muss. Nach erfolgreichem Aufruf und Eingabe des Passwortes sollte ein Prompt wie etwa

```
theseus$
```

zu sehen sein. Ihr befindet euch in eurem Heimatverzeichnis auf unserem Server `theseus` und arbeitet über eine verschlüsselte Verbindung.

- Durch Eingabe des Kommandos `pwd` könnt ihr euch jederzeit anzeigen lassen in welchem Verzeichnis ihr euch befindet. Zu Beginn etwa `/home/login`.
- Mit dem Kommando `ls` könnt ihr euch die Dateien und Ordner im aktuellen Verzeichnis ausgeben lassen. Gebt ihr ls zusätzlich die Argumente `-lh` an, also `ls -lh`, bekommt ihr eine ausführlichere Darstellung des Ordnerinhalts (`-l` für Listendarstellung und `-h` für human readable Größenangaben).
- Das Kommando `cd` dient zur Navigation durch die Ordner. Ohne weitere Argumente landet ihr wieder in eurem Heimatverzeichnis. Gebt ihr einen Ordnernamen an, wechselt ihr in diesen Ordner (danach einfach mit `pwd` den Unterschied zu vorher ansehen). Das Argument `.` steht für den aktuellen Ordner und das Argument `..` für den übergeordneten Ordner (`cd ..` wechselt also in den Ordner in dem der aktuelle Ordner enthalten ist).
- Mit dem Programm `mkdir` könnt ihr Ordner anlegen (z.B. `mkdir testOrdner`).
- Das Kommando `rm` erlaubt euch Dinge zu löschen (ACHTUNG: `rm` arbeitet entgültig!!!). Mit der Option `-r` können auch Ordner gelöscht werden (z.B. `rm -r testOrdner`).

- `mv` verschiebt Dateien/Ordner bzw. nennt diese um und `cp` kopiert (`cp sourceFile targetFile`).
- Ordner werden durch `/` forward slashs getrennt, z.B. um eine Datei in den Ordner `tmpOrdner` zwei Ordner weiter oben in der Dateihierarchie zu verschieben

  ```
  mv sourceFile ../../tmpOrdner/newFileName
  ```

- Das Programm `man` zeigt euch Hilfeseiten zu den diversen Kommandos an. Mit `man cp` landet ihr z.B. bei der Hilfeseite zu `cp`. Mit dem Buchstaben `q` schließt ihr die Hilfe, mit `Ctrl-f` bzw. `Ctrl-b` könnt ihr einen Screen voll nach vorne bzw. zurück blättern. Wenn ihr die Hilfeseite seht, könnt ihr darin suchen indem ihr `/Suchbegriff` eingebt, sucht z.B. in `man cp` nach `-r` (gebt „`/-r`“ Enter ein). Mit dem Buchstaben klein `n` bzw. groß `N` springt ihr zum nächsten Treffer bzw. zum vorherigen.

Aufgabe: Macht euch mit der Shell vertraut, übt den Umgang mit den kurz vorgestellten Programmen und Kommandos und schlagt im Internet nach, wenn euch etwas fehlt. Seht nach, was Tab-Completion ist (sehr nützlich!).

2 Vim

Um später unsere Dateien zu schreiben, benötigen wir einen Texteditor. Bestimmt sind einige bekannt—vlt. auch welche die über eine ssh Verbindung funktionieren. Wir empfehlen sich mit einem Shell-Editor ein wenig auseinanderzusetzen.

Exemplarisch sei hier eine kurze Einführung in `vim` gegeben:

`vim` wird wie die obigen Programme in der Shell durch Eingabe des Namens aufgerufen. Ungewohnt ist vermutlich zunächst folgendes Prinzip: Es gibt zwei Modi:

- Kommando Modus
- Eingabe Modus.

Im Kommando-Modus lässt sich z.B. das Dokument speichern, der Editor beenden und effizient navigieren (hier funktioniert z.B. auch `Ctrl-f` bzw. `Ctrl-b` um Screenweise zu blättern). Mit Eingabe von `Esc` bzw. `Ctrl-c` kommt ihr jeweils in den Kommando-Modus zurück. Kommandos die mit einem `:` beginnen mit Enter bestätigen oder mit `Esc` oder `Ctrl-c` abbrechen:

- `:w` speichert das Dokument - falls noch kein Dateiname angegeben wurde einfach `:w DateiName` eingeben (Endungen sind ganz normaler Bestandteil des Dateinamens und nicht notwendig! Ihr könnt prinzipiell auch in eine Datei `meinFilm.avi` speichern. Der Inhalt wird aber dennoch normaler Text sein.).
- `:q` beendet vim. Falls ihr noch nicht gespeichert habt und dennoch beenden wollt gebt `:q!` ein.
- Eingabe von `u` bedeutet undo, `Ctrl-r` redo.
- Die Buchstaben `h,j,k,l` können im Kommando Modus ebenso wie die Pfeiltasten $\leftarrow, \downarrow, \uparrow$ und $\rightarrow$ zur Navigation verwendet werden.

- Eingabe von `dd` löscht eine Zeile, gebt ihr `13dd` ein, so löscht ihr 13 Zeilen.

In den Eingabe Modus gelangt man z.B. durch Eingabe des Buchstabens `i` oder `a` (was ist der Unterschied?). Navigation ist im Eingabe Modus mit den Pfeiltasten möglich. Zum löschen von Text kann `Entf/Del` verwendet werden, Eingabe durch einfaches Schreiben.

Aufgabe: Nutzt einen Shell-Editor eures Vertrauens (z.B. `vim`), macht euch mit diesem vertraut (schlagt evtl. im Internet weitere Kommandos nach) und erstellt eine Datei „`hello.cpp`“ mit folgendem Inhalt in einem neuen Ordner Namens „`meinErstesCppProgramm`“ (bitte abschreiben).

```
#include <iostream>

int main(int argc, char ** argv)
{
  std::cout << "Hello world!" << std::endl;
  return 0;
}
```

3 Hello World!

In der letzten Aufgabe habt ihr die Datei „`hello.cpp`“ in dem sonst leeren Ordner Namens „`meinErstesCppProgramm`“ erstellt. Wie euch sicherlich aufgefallen ist, ist dies ein erstes C++ Programm. Lest dieses nochmals auf Tippfehler hin durch.

Nun wird es Zeit euer erstes C++Programm zu kompilieren (nehmt hierzu das Programm `g++` (ein C++ Compiler) mit eurem Dateinamen als Argument):

```
g++ hello.cpp
```

Wenn alles gut ging habt ihr keine weitere Ausgabe als den neuen Prompt bekommen. Keine Angst—No news is good news! Das ist zumindest der Standard unter Shell Programmen. Falls ihr Fehler habt lest den Code nochmals auf Fehler hin durch und probiert es erneut. Falls ihr keine Fehler habt müsste eine Datei Namens `a.out` generiert worden sein (mit `ls` lässt sich dies schnell überprüfen).

Diese beinhaltet den kompilierten Code aus eurer Datei, der vom Computer ausgeführt werden kann. Ruft dazu euer Programm auf indem ihr

```
./a.out
```

eingebt (hier taucht auch wieder der `.` auf, der für das aktuelle Verzeichnis steht). Ihr müsstet folgendes sehen:

```
theseus$ ./a.out
Hello world!
theseus$
```


Herzlichen Glückwunsch! Ihr habt ein C++ Programm geschrieben, kompiliert und ausgeführt.

Spielt ein wenig mit dem Code, seht nach, was passiert wenn ihr einen Strichpunkt oder Ähnliches vergesst und versucht eine einfache `for`-Schleife zu verwenden. Ihr könnt eurem Programm übrigens auch gleich einen Namen geben indem ihr die Option `-o meinExecutableName` des `g++` Compilers verwendet:

```
theseus$ rm a.out
theseus$ g++ -o hello hello.cpp
theseus$ ./hello
Hello world!
theseus$
```

4 Submission

Für alle die uns gerne eine Lösung zukommen lassen möchten (z.B. für einen Schein) bieten wir für bestimmte Dateien ein submission System auf theseus an (nachfolgendes Kommando geht nur, wenn ihr auf unseren Rechnern seid). Ihr müsst hierfür im SLC `https://slcbeta.mathematik.uni-ulm.de:8443/login.html` angemeldet sein. Für dieses Blatt könnt ihr dies mit dem Kommando

```
submit cpp 1 hello.cpp
```

erledigen, wobei ihr zuvor natürlich in das Verzeichnis geht in dem die Datei hello.cpp ist. Die 1 ersetzt ihr bei den weiteren Blättern durch die jeweilige Nummer. Möchtet ihr als Team abgeben (maximal 2 Personen—zur Not 3) erstellt ihr im gleichen Verzeichnis eine Datei `team` in der pro Zeile genau ein login-Name und sonst nichts steht (diese Personen müssen ebenfalls im SLC angemeldet sein) und submitted eure Lösung wie folgt:

```
submit cpp 1 hello.cpp team
```

5 Funktionen

Da ihr in den Übungen schon einen sehr wissenden Eindruck gemacht habt, hier noch etwas zu Funktionen: Ladet euch die Datei `blatt01.tgz` herunter (z.B. mit dem Programm `wget` und der url in Anführungszeichen als Argument `http://www.mathematik.uni-ulm.de/sai/ws10/cpp/blaetter/blatt01.tgz`). Entpackt diese mit dem Programm `tar` mit den Argumenten `-xzf` und `blatt01.tgz` in euer aktuelles Verzeichnis. Ihr solltet nun drei Dateien in eurem Ordner haben `hello.cpp`, `myFunctions.h` und `myFunctions.cpp`. Seht euch deren Inhalt an und macht euch das Zusammenspiel bewusst. `myFunctions.h` beinhaltet zwei Funktionsdeklarationen, `myFunctions.cpp` deren Implementierung und `hello.cpp` nutzt die Funktionsdeklarationen ohne von deren Implementierung/Definition zu wissen.

Die beiden Einheiten werden separat kompiliert (`-c` Option von `g++` erzeugt die Object-Files mit Endung `.o`) und anschließend in der Executable `a.out` zusammengeführt:

```
theseus$ls
blatt01.tgz       hello.cpp         myFunctions.cpp   myFunctions.h
theseus$g++ -c hello.cpp
theseus$g++ -c myFunctions.cpp
theseus$ls
blatt01.tgz       hello.cpp         hello.o           myFunctions.cpp
myFunctions.h     myFunctions.o
theseus$g++ hello.o myFunctions.o
theseus$ls
a.out             blatt01.tgz       hello.cpp         hello.o
myFunctions.cpp   myFunctions.h     myFunctions.o
theseus$./a.out
Hello World!
result=9
meine zweite Funktion wurde aufgerufen
theseus$
```

Zeichnet euch die Beziehung der drei Dateien auf, schreibt eure eigene Header-Datei mit mindestens einer Funktion und ruft diese in der Methode `main` auf.

Wenn ihr an irgendeiner Stelle auf Probleme trefft, die ihr nicht lösen könnt, schreibt mir dies bitte vor der nächsten Übung, damit ich darauf noch etwas eingehen kann.

6 FAQ

- Werden alle unsere Übungsblätter so textlastig?
- Nein! Nur zu Beginn gilt es einen gemeinsamen Stand zu schaffen. Versucht diese grundlegenden Dinge zu beherrschen!

Viel Spaß!