Programmierkurs
Dr. Steffen Müthing
IWR, Universität Heidelberg

Aufgabenblatt 1
Abgabe 29. Oktober 2018
ipk-exercises:2018-ws-3-gc8df90e

Aufgabenblatt 1

Allgemeine Hinweise:

- Lesen Sie sich bitte zuerst die ausliegende Nutzungsordnung des Pools durch!
- Das erste Übungsblatt ist dazu gedacht, während der Übung bearbeitet zu werden. Auf diesem Übungsblatt müssen Sie nächste Woche nichts ankreuzen.
- Beginnend mit dem nächsten Übungsblatt sind die Übungsblätter nur noch online unter https://conan.iwr.uni-heidelberg.de/teaching/ipk_ws2018/ verfügbar, dann jeweils am späten Freitagnachmittag.
- Während der Übung erhalten Sie Unterstützung vom Dozent und den Tutoren. Nutzen Sie diese Möglichkeit und melden Sie sich bei Fragen!
- Wenn Sie die virtuelle Maschine auf Ihrem Notebook verwenden wollen und bei der Installation noch Hilfe brauchen, wenden Sie sich während der Übungen an uns.
- Unter https://conan.iwr.uni-heidelberg.de/teaching/ipk_ws2018/ finden Sie die Folien der ersten Vorlesung.

Aufgabe 1

Diese Aufgabe soll Ihnen dabei helfen, sich an der Kommandozeile zurechtzufinden.

(a) Benutzen Sie die Kommandozeile, um unter Verwendung der in der Vorlesung vorgestellten Befehle `cd` und `mkdir` in Ihrem Home-Verzeichnis ein Verzeichnis `uebungen` und darin ein Verzeichnis `uebung01` anzulegen.

```
~ $ mkdir uebungen
~ $ cd uebungen
uebungen $ mkdir uebung01
uebungen $ cd uebung01
uebung01 $
```

Wenn Sie nähere Informationen über einen Befehl möchten, schauen Sie sich die Dokumentation des Befehls, die sogenannte *manpage* mit dem Befehl `man BEFEHL` an, z.B. `man mkdir`. Sie können in der Ausgabe mit den Pfeiltasten scrollen und mit der Taste `"q"` zur Kommandozeile zurückkehren.

(b) Erstellen Sie in diesem Verzeichnis mit einem Texteditor Ihrer Wahl die Datei `helloworld.cc` mit folgendem Inhalt:

```cpp
#include <iostream>

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

Kompilieren Sie das Programm und führen Sie es aus. Zum Kompilieren verwenden wir hier den in der Vorlesung beschriebenen Wrapper `ipkc++` für den Clang-Compiler, die Option `"-o <name>"`

sagt dem Compiler, wie das erzeugte Programm heißen soll.

```
uebung01 $ ipkc++ -o helloworld helloworld.cc
uebung01 $ ./helloworld
Hello world!
uebung01 $
```

Hinweis: Wenn Sie direkt auf Ihrem Notebook arbeiten (nicht an einem Pool-Rechner oder in der virtuellen Maschine), ersetzen Sie `ipkc++` durch `"g++ -Wall"` oder `"clang++ -Wall"`. Sie benötigen einen ausreichend neuen Compiler. Dies können Sie testen, indem Sie folgendes Programm in einer Datei speichern und versuchen, die Datei zu kompilieren:

```
#include <iostream>

auto message()
{
  return "Hello world!";
}

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

Falls Ihr Compiler hierbei einen Fehler erzeugt, melden Sie sich bitte bei uns.

(c) Probieren Sie aus, was passiert, wenn Sie die Option `"-o <name>"` weglassen. Zur Erinnerung: Sie können den Inhalt des aktuellen Verzeichnisses mit dem Befehl `ls` anzeigen. Versuchen Sie, das vom erzeugte Programm ohne ein vorangestelltes `"./"` auszuführen.

(d) Geben Sie die Datei `helloworld.cc` mit Hilfe des Programms `cat` auf der Kommandozeile aus. Was passiert, wenn Sie das gleiche mit dem kompilierten Program `helloworld` versuchen?

(e) Suchen Sie mit Hilfe von `grep` in der Datei `/usr/include/stdio.h` nach dem Begriff *FILE*. Probieren Sie auch die Option `"-n"` aus.

(f) Suchen Sie mit Hilfe von `grep` in der Datei `helloworld.cc` nach *Hello world*. Was beobachten Sie?

(g) Die Shell trennt Argumente, sobald ein Leerzeichen auftaucht. Sie können dies umgehen, indem Sie Argumente, die aus mehreren Wörtern bestehen, in Anführungszeichen setzen. Versuchen Sie, den Befehl aus dem vorhergehenden Aufgabenteil so zum Laufen zu bringen.

Aufgabe 2

Diese Aufgaben sind für die Fortgeschritteneren unter Ihnen, die sich schon mit der Shell auskennen. Um diese Aufgaben zu lösen, müssen Sie eventuell die *man pages* der Befehle lesen und / oder im Internet nach Tips suchen.

(a) Finden Sie mit Hilfe der Shell, Pipes und den Befehlen `find` und `wc` heraus, wie viele C-Header-Dateien (Endung `.h`) sich im Verzeichnis `/usr/include` und allen Unterverzeichnissen befinden.

(b) Finden Sie mit Hilfe der obigen Befehle sowie `xargs` und `grep` mit einer passenden *regular expression* heraus, wie viele *include statements* sich in diesen Dateien befinden. Include statements bestehen aus einer Zeile, die mit `#include` beginnt, allerdings können vor dem `#include` noch beliebig viele Leerzeichen stehen.