Programmierkurs IPK WS21/22
Lorenz Braun
ZITI, Universität Heidelberg

Aufgabenblatt 1
Abgabedatum —
ipk-exercises:2021-ws-2-g2af2023

Aufgabenblatt 1

Allgemeine Hinweise:

- Dieser Zettel muss nicht abgegeben werden. Stattdessen wird dieser Zettel als Präsenzaufgabe während der ersten Übung bearbeitet. Die letzte Aufgabe ist keine richtige Aufgabe, stattdessen wird dort das Votiersystem erklärt.
- Wir gehen bei den Aufgabe dieser Vorlesung davon aus, dass Sie Zugriff auf eine UNIX-Umgebung haben. Für Hinweise zum Einrichten einer solchen Programmierumgebung besuchen Sie bitte den Moodle Kurs: https://moodle.uni-heidelberg.de/course/view.php?id=8916
- Alle Folien und Übungen werden unter https://moodle.uni-heidelberg.de/course/view.php?id=8916 veröffentlicht.
- Ab der zweiten Woche wird jeweils Freitags bis 12 Uhr der nächste Zettel in Moodle veröffentlicht und Sie haben eine Woche Zeit (bis Freitag 12 Uhr) eine Abgabe einzureichen. Details dazu entnehmen Sie bitte den Hinweisen auf dem entsprechendem Zettel.
- Die Bewertung Ihrer zukünftigen Abgaben geschieht durch ein Votiersystem, welches in Aufgabe 4 erklärt wird.
- Legen Sie sich für jede Übung einen eigenen Ordner an (z.B. `uebung01`).
- Abgaben werden in Gruppen von drei Personen gemacht (soweit das mit der Teilnehmeranzahl möglich ist).

Aufgabe 1: Kommandozeile

(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.

```
1  ~ $ mkdir uebungen
2  ~ $ cd uebungen
3  uebungen $ mkdir uebung01
4  uebungen $ cd uebung01
5  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 Gnu compiler `g++`, die Option `"-o <name>"` sagt dem Compiler, wie das erzeugte Programm heißen soll.

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

Hinweis: Es ist sinnvoll Warnmeldungen beim Compilieren zu aktivieren. Ersetzen sie dafür `"g++"` durch `"g++ -Wall"`. Für den Fall, dass Sie den `clang` compiler verwenden können sie entsprechend `"clang++ -Wall"` benutzen.

(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: Syntaxfehler und Compiler-Meldungen

In dieser Aufgabe arbeiten wir mit C++-Quellcode, aber Sie müssen diesen nicht wirklich verstehen. Es geht hier primär darum, die Fehlermeldungen des Compilers zu verstehen und die Tipps zum Beheben der Fehler richtig anzuwenden.

(a) Erstellen Sie die Datei `errors.cc` mit folgendem (absichtlich fehlerhaften) Inhalt:

```cpp
#include <iostream>

int main(int argc, char** argv)
  stf::cout << "Typing is difficult" << endl;
  int ret = 0;
  return retv
}
```

Versuchen Sie, das Programm zu kompilieren. Der Compiler wird diverse Fehlermeldungen ausgeben. Beheben Sie den oder die angezeigten Fehler und starten Sie den Compiler erneut. Dies kann unter Umständen dazu führen, dass der Compiler andere Fehler anzeigt. Machen Sie so lange weiter, bis das Programm übersetzt werden kann.

(b) Erstellen Sie die Datei `legalbutwrong.cc` mit folgendem Inhalt:

```cpp
#include <iostream>

int main(int argc, char** argv)
{
  int n = 10;
  // calculate the sum of all numbers from 1 to n
  int i;
  int sum = 0;
  for (int j = 1 ; i <= n ; j = j+1)
  {
    sum = sum + j;
  }
  std::cout << sum << std::endl;
  return 0;
}
```

Kompilieren Sie das Programm und führen Sie es aus. Falls das Programm "hängt", können Sie es mit der Tastenkombination "CTRL-C" beenden.

Das Programm ist zwar syntaktisch korrekt, aber es hat einen Fehler. Der Compiler kann Ihnen oft helfen, solche Probleme aufzuspüren. Hierzu müssen Sie ihn anweisen, Ihnen nicht nur Fehler, sondern auch Warnungen anzuzeigen. Sowohl GCC als auch clang benötigen hierfür die Option "-Wall" (warn all). Bei ipkc++ ist diese Option bereits standardmässig eingeschaltet.

Kompilieren Sie das Programm erneut mit der zusätzlichen Option "-Wall" und beheben Sie die vom Compiler gemeldeten Probleme. Das Programm sollte nun die richtige Lösung (55) ausgeben.

Aufgabe 3: Kommandozeile für Fortgeschrittene

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.

Aufgabe 4: Votiersystem

Anstatt alle Abgaben zu korrigieren und Punkte zu verteilen arbeiten wir mit einem Votiersystem. Das funktioniert folgendermaßen:

- Legen Sie in Ihrer Abgabe eine Datei `votieren.txt` an. In dieser Textdatei beschreiben Sie welche Aufgabe Sie lösen konnten und welche Sie im Tutorium präsentieren können.

  Beispiel:

```
Martin: Aufgabe 1, 2, 3
Julia: Aufgabe 2
Peter: Aufgabe 1 und 2
```

- Ihre Lösung muss nicht perfekt sein. Wenn Sie denken, dass Sie die Aufgabe größtenteils gelöst haben und den korrekten Lösungsansatz haben dürfen Sie für diese votieren.
- Auf späteren Aufgabenzetteln wird es Votierpunkte für Teilaufgaben geben. Das wird jeweils klar gekennzeichnet und Sie geben für jeden Votierpunkt an, ob Sie diesen Teil gelöst haben.

- Es kann sein, dass Sie Ihre Lösung (oder einen Teil davon) im Tutorium vorstellen müssen.

In der Übung können die folgenden Szenarien eintreten:

- Sie stellen erfolgreich einen Teil der Lösung vor. In diesem Fall bekommen Sie alle Punkte, für die Sie in diesem Blatt votiert haben.
- Sie müssen nichts vorstellen. Auch in diesem Fall bekommen Sie alle Punkte, für die Sie votiert haben.
- Sie müssen vorstellen, sind aber nicht dazu in der Lage. In diesem Fall werden Ihnen alle Punkte für dieses Blatt aberkannt. Das betrifft jedoch nicht Ihre Übungsgruppenpartner.

Dabei überschreiben spätere Regeln die vorherigen (falls Sie mehrere Aufgaben vorstellen).

Jetzt gibt es noch zwei Spezialfälle:

- Ihre Gruppe votiert für Aufgaben, die Sie offensichtlich nicht gelöst haben. In diesem Fall kann der Tutor beschließen für diese Aufgabe keine Punkte an die Abgabegruppe zu verteilen. Sollte dies häufig vorkommen wird der Tutor Sie darauf ansprechen.
- Sollte Ihre Abgabe von einer anderen Gruppe kopiert sein, kann der Tutor entscheiden beiden Gruppen die Punkte dieser Aufgabe oder die Punkte des gesamten Blatts abzuerkennen. Natürlich ist es im Zweifel schwierig festzustellen, ob es sich um Kopien handelt und wir werden verantwortungsvoll mit dieser Regelung umgehen und erst das persönliche Gespräch suchen.

Um die Klausurzulassung zu erreichen benötigen Sie 50% der erreichbaren Votierpunkte. Es wird Bonusaufgaben geben, durch die Sie ihre Punktzahl verbessern können. Diese werden nicht auf die Zahl der “erreichbaren Punkte” dazu addiert.