

Einführung

Einführung in die Programmierung

Michael Felderer (QE)

Institut für Informatik, Universität Innsbruck

Überblick

- Programmierung allgemein
- C-Hintergrund
- Das erste C-Programm

PROGRAMMIERUNG ALLGEMEIN

Maschinensprache

- Ein Computer ist eine Maschine, die ein Problem löst, indem sie Befehle (instructions) ausführt.
- Programm = Satz von Befehlen, der zur Ausführung einer bestimmten Aufgabe zusammengestellt wurde.
- Die elektronischen Schaltungen eines Computers „kennen“ nur eine begrenzte Menge einfacher Maschinenbefehle.
- **Alle Programme** müssen in solche dem Rechner bekannte Maschinenbefehle umgewandelt werden, ehe sie ausführbar sind.
- Beispiel (MIPS 32-Bit Architektur, Befehl als Bitmuster):
 - 00000001000010010101000000100000
 - Addiere binäre Werte in Register 8 und 9 und speichere Ergebnis in Register 10
 - add \$10, \$8, \$9

Höhere Programmiersprachen

- Maschinenbefehle bilden eine Sprache, die ein Computer verarbeiten kann, die Maschinensprache (machine language).
 - Unterschiedliche Computer (CPUs) können unterschiedliche Maschinensprachen haben!
- Menschen können Maschinensprachen nur für sehr kleine Programme benutzen und daher hat man **höhere Programmiersprachen** entwickelt, die das Programmieren erheblich erleichtern.
- Da die Maschine wiederum die Befehle der höheren Programmiersprache nicht kennt, muss man diese dann in die Maschinensprache **übersetzen**.

Übersetzen – Compilieren

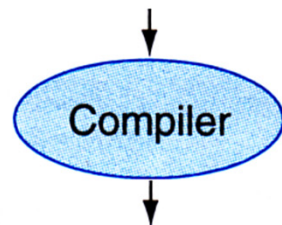
- Beim Übersetzen wird jeder Befehl des in der höheren Programmiersprache geschriebenen Programms vor dem Ablauf in eine entsprechende **Folge von Maschinenbefehlen** übersetzt und in einer Datei gespeichert.
- Ausgeführt wird dann das Maschinenprogramm aus dieser Datei.
- Das Programm, das für die Übersetzung zuständig ist, wird allgemein **Compiler** genannt.
 - In der Realität setzt sich der Übersetzungsvorgang aus mehreren Stufen zusammen, und es werden noch zusätzliche Programme benötigt.

Beispiel – C und MIPS-Prozessor



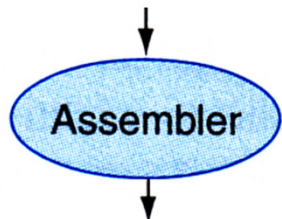
Programm in
einer höheren
Programmiersprache
(in C)

```
swap(int v[], int k)
{int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```



Programm in
Assemblersprache
(für MIPS)

```
swap:
    muli $2, $5, 4
    add  $2, $4, $2
    lw   $15, 0($2)
    lw   $16, 4($2)
    sw   $16, 0($2)
    sw   $15, 4($2)
    jr   $31
```



Programm in bi-
närer Maschinen-
sprache
(für MIPS)

```
000000001010000100000000000011000
000000000000110000001100000100001
100011000110001000000000000000000
100011001111001000000000000000100
101011001111001000000000000000000
101011000110001000000000000000100
00000011111000000000000000001000
```



- In einem Maschinenprogramm muss der Programmierer dem Computer jeden auszuführenden Schritt genau vorschreiben.
 - Sehr zeitaufwändig und fehleranfällig!
- Ein Programmierer sollte sich mehr auf das Lösen des eigentlichen Anwendungsproblems konzentrieren.
- Dazu entstanden Ende der 1950er Jahre die ersten höheren Programmiersprachen:
 - Diese sollten es ermöglichen, eine Problemlösung in einer eher fachspezifischen Notation anzugeben.
 - Höhere Programmiersprachen sollen die Umsetzung problemorientierter Algorithmen erleichtern und werden auch als problemorientierte Programmiersprachen bezeichnet.
 - Heute gibt es viele Programmiersprachen.
 - Siehe http://en.wikipedia.org/wiki/List_of_programming_languages



C - HINTERGRUND



- Woher stammt C?
 - AT&T Bell Laboratories
 - 1969 erste C Version von Dennis Ritchie und Ken Thompson für Unix entwickelt
- Versionen
 - 1978 – *K&R-C* (Brian W. Kernighan und Dennis Ritchie) in der ersten Auflage des Buches *The C Programming Language*.
 - 1989 – C89 Standard (ANSI C, Standard C)
 - 1990 – ANSI C von ISO übernommen (C 90)
 - 1999 – C 99 Standard
 - Rückwärtskompatibel
 - Wird nicht von allen Compilern komplett unterstützt!
 - Wird in dieser LV verwendet.
 - 2011 – C11 Standard



- Imperative Programmiersprache
- Wenige Schlüsselwörter (reservierte Wörter)
- Direkte Speicherzugriffe und sehr hardwarenahe Programmierung
- Schwach ausgeprägtes Modulkonzept
- Trotz weniger Schlüsselwörter sehr mächtige Sprache
 - **Vor- und Nachteil!**



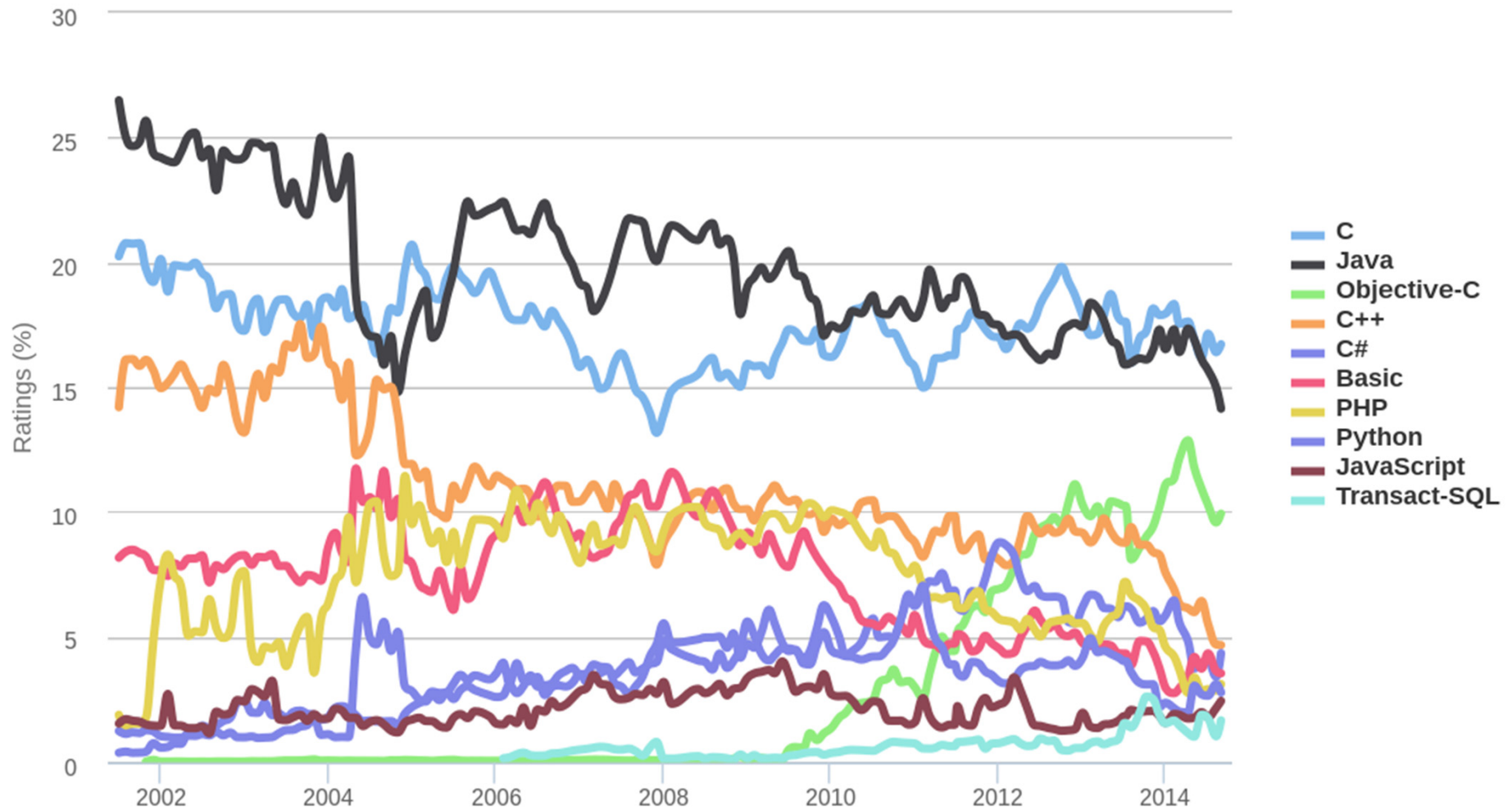
- Wo wird C heute verwendet?
 - Betriebssysteme (z.B. Linux) und Systemprogrammierung
 - Hardwarenahe Programmierung
 - Programmierung eingebetteter Systeme
 - Spieleprogrammierung
 -
- C ist die Grundlage weiterer Programmiersprachen.
 - C++ - Objektorientierte Variante von C.
 - Objective C – Objektorientierte Erweiterung von C (Mac OS).
 - Weitere Sprachen, die von C beeinflusst wurden:
 - Java, C#, Perl, Python, R ...
- C-Compiler sind auf fast allen Systemen verfügbar.

TIOBE-Index (September 2014)

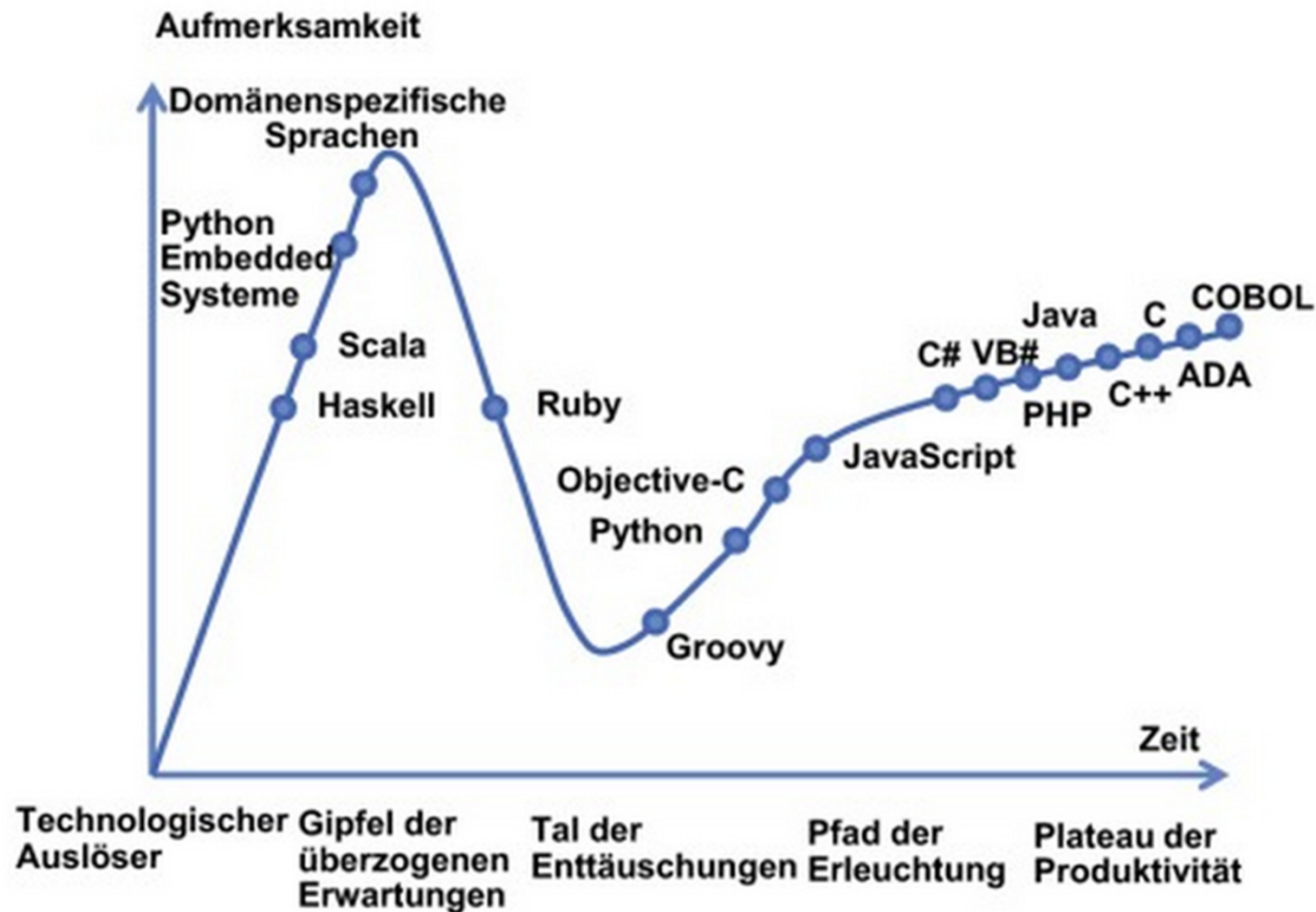


TIOBE Programming Community Index

Source: www.tiobe.com



Technology Hype Cycle für Programmiersprachen



DAS ERSTE C-PROGRAMM

Das erste C-Programm (C99-Stil)

```
1.  /* Das erste C-Programm */

    /* Einbinden von Header-Dateien, die wichtige
       * Informationen beinhalten */
2.  #include <stdio.h>          // für Funktion printf
    #include <stdlib.h>        // für EXIT_SUCCESS

    /* Bei main beginnt unser Programm
       * Zuerst wird "Hello World!" auf die Konsole ausgegeben,
       * danach wird das Programm sauber beendet.*/
3.  int main(void){
4.      printf("Hello World!\n");
5.      return EXIT_SUCCESS;
    }
```

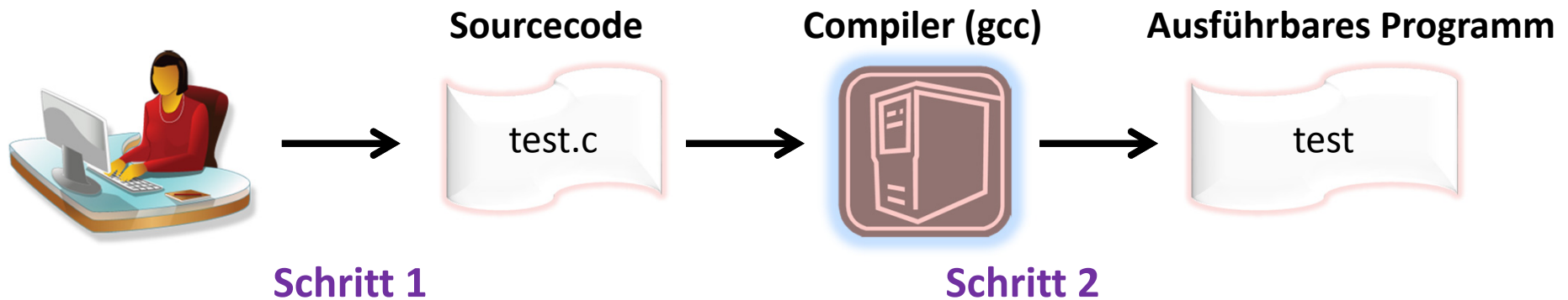
Linux Kommandozeile (Programm hat den Namen test.c):

```
[...]$ gcc -Wall -Werror -std=c99 test.c -o test
[...]$ ./test
Hello World!
```


C-Programmierung

- Grundsätzlich gibt es 2 Arten von Dateien
 - Quelltextdateien
 - Enthalten den Source-Code.
 - Headerdateien
 - Beinhalten in der Regel Informationen, die man für die Übersetzung benötigt.
 - Beinhalten Informationen über neu definierte Datentypen.
- Source-Code
 - Programm, das in diesem Fall in der Sprache C geschrieben wurde.
 - Kommentare, die den Inhalt des Programms sinnvoll beschreiben.
 - Diese Kommentare werden nicht in Maschinsprache übersetzt!

Einfache Programmerzeugung (nur eine Source-Datei)



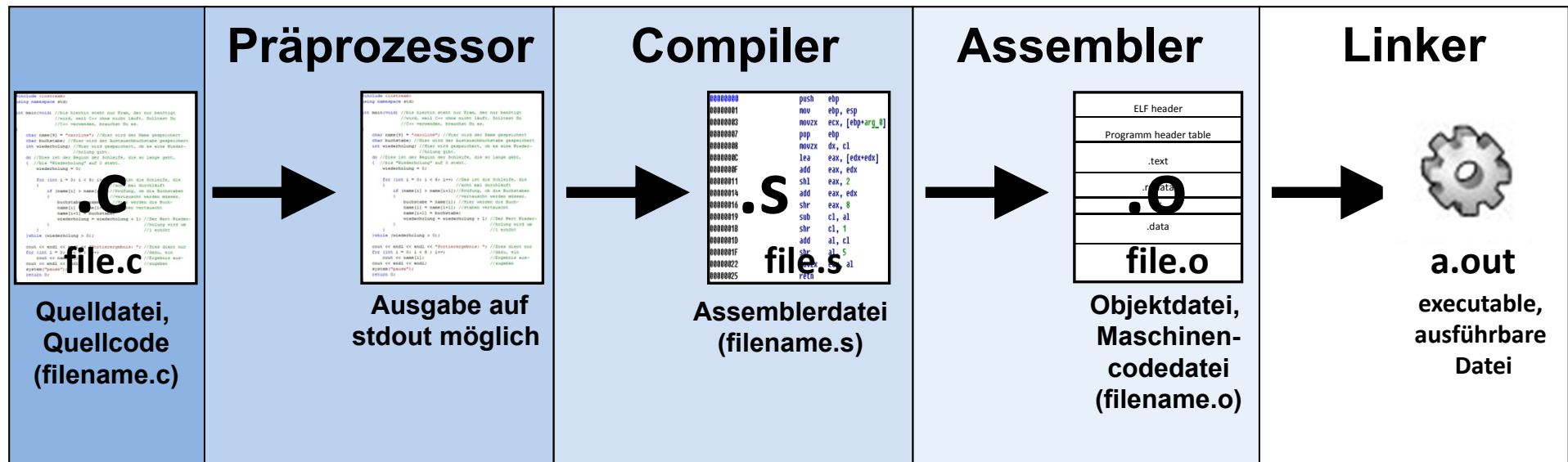
- Schritt 1
 - Erstellen des Programms in einem beliebigen Editor (z.B. kate, gedit, nano)
 - Abspeichern unter einem beliebigen Namen:
 - Der Name sollte aber Sinn ergeben und auf **.c** enden (z.B. **test.c**).
- Schritt 2 (sehr vereinfacht!)
 - Übersetzen des Programms auf der Kommandozeile.
 - Beispiel für das obige Programm auf einem ZID-Rechner:
gcc test.c -o test (oder **cc test.c -o test**)
 - Erzeugt das ausführbare Programm **test**.

- Compiler (und noch mehr!)
 - Für unterschiedliche Betriebssysteme und Architekturen erhältlich.
 - In den meisten Linux-Distributionen enthalten.
- Einfacher Aufruf
 - `gcc program.c`
 - Erzeugt ausführbares Programm **a.out** (assembler output) unter Linux.
 - Unter Windows: **a.exe** (z.B. von MinGW erzeugt).
- Aufruf mit Output-Datei
 - `gcc program.c -o program`
- Kompliziertere Aufrufe
 - Mehrere Dateien, Optimierung etc.
 - Aufrufkonvention für diese Vorlesung und die Übung (ZID-Rechner):
`gcc -Wall -Werror -std=c99 program.c -o program`

C-Umgebungen in dieser Vorlesung

- Die meisten Programme wurden am `zid-gpl.uibk.ac.at` getestet und sollten daher auf ZID-Rechnern ohne Warnungen und Fehler lauffähig sein.
 - Falls zu Demonstrationszwecken absichtlich fehlerhafter Code verwendet wurde, dann wird darauf hingewiesen.
- Einige Ausgaben auf den Folien wurden mit Eclipse CDT + MinGW auf einem Windows 7 Laptop erzeugt.
 - Eclipse CDT: <http://www.eclipse.org/cdt/>
 - MinWG: <http://www.mingw.org/>

Und was passiert da wirklich im Hintergrund?



- Diese Teile werden im Laufe der Vorlesung noch ausführlich besprochen.

Das erste C-Programm (C99-Stil)

```
1.  /* Das erste C-Programm */

    /* Einbinden von Header-Dateien, die wichtige
       * Informationen beinhalten */
2.  #include <stdio.h>          // für Funktion printf
    #include <stdlib.h>        // für EXIT_SUCCESS

    /* Bei main beginnt unser Programm
       * Zuerst wird "Hello World!" auf die Konsole ausgegeben,
       * danach wird das Programm sauber beendet.*/
3.  int main(void){
4.      printf("Hello World!\n");
5.      return EXIT_SUCCESS;
    }
```

Linux Kommandozeile (Programm hat den Namen test.c):

```
[...]$ gcc -Wall -Werror -std=c99 test.c -o test
[...]$ ./test
Hello World!
```

Das erste C-Programm (Erklärung zu 1.)

```
/* Das erste C-Programm */  
/* Einbinden von Header-Dateien, die wichtige  
 * Informationen beinhalten */
```

- Hier stehen Kommentare.
- Damit wird der Programmcode kommentiert!
- Kommentare enthalten keinen ausführbaren Code!
- Es existieren zwei Formen
 - `/* ... */` Einzeilig oder mehrzeilig
 - `//.....` Einzeilig (eigentlich nur in C99, gcc kennt solche Kommentare auch ohne C99)

Das erste C-Programm (Erklärung zu 2.)

```
#include <stdio.h> // für Funktion printf  
#include <stdlib.h> // für EXIT_SUCCESS
```

- Präprozessordirektiven (beginnen mit #)
- Diese Direktiven werden vom Präprozessor verarbeitet.
 - Der Präprozessor bearbeitet das gegebene Programm (durch reine Textersetzung) und übergibt das Ergebnis an den eigentlichen Compiler.
- Die **#include**-Direktive dient zum Einbinden von Dateien.
 - Der Inhalt dieser Dateien wird vom Präprozessor an dieser Stelle eingesetzt!
- In diesem Fall wird zum Beispiel die Datei **stdio.h** eingebunden, die Informationen über den Befehl **printf** beinhaltet.

Das erste C-Programm (Erklärung zu 3.)

```
int main(void)
```

- Hier beginnt das eigentliche Programm.
- **main()** ist eine Funktion, die in jedem Programm enthalten sein muss.
 - Ein Programm kann aus mehreren Funktionen bestehen.
- Das ist der Einstiegspunkt für jedes C-Programm.
- In diesem Fall werden keine Daten übergeben, daher **(void)** und es wird etwas zurückgegeben (vom Datentyp **int** – wird noch ausführlich erklärt).
- Die Anweisungen des Programms stehen zwischen { und }.

Das erste C-Programm (Erklärung zu 4.)

```
printf("Hello World!\n");
```

- An dieser Stelle wird die Funktion **printf** aufgerufen und die Zeichenkette **Hello World!** übergeben (alle Zeichen zwischen den Anführungszeichen).
 - Am Ende wird ein „\n“ hinzugefügt und damit wird ein Zeilenumbruch erzwungen.
- Dieser Aufruf wird mit einem Semikolon abgeschlossen!
- Resultat bei Ausführung: Die Zeichenkette wird auf den Bildschirm (Konsole) ausgegeben.

Das erste C-Programm (Erklärung zu 5.)

return EXIT_SUCCESS;

- Rückgabe eines Wertes (wird noch ausführlich erklärt).
- Hier könnte 0 (**return 0;**) zurückgegeben werden.
 - EXIT_SUCCESS ist sauberer.

Bienenkorb/Murmelgruppe

- Didaktische Methode, die auch in großen Gruppen funktioniert
- Kleinstgruppen von 2-4 Studenten
- Lösung einer kleinen, kurzen Aufgabe in der Gruppe
- Zeit zwischen 5 und 10 Minuten
- Sammlung von Lösungen

Aufgabe

- Wo genau liegt der Fehler im folgenden C-Programm?

```
int main(void) {  
    printf("Was ist hier falsch?\n");  
    printf("Es fehlt was!\n");  
    return 0;  
}
```

Welche Zeichen darf man in C verwenden?

- Buchstaben
 - A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
 - a b c d e f g h i j k l m n o p q r s t u v w x y z
- Dezimalziffern
 - 0 1 2 3 4 5 6 7 8 9
- 29 Grafiksymbole
 - ! " % & / () [] { } \ ? = ' # + * ~ - _ . : ; , | < > ^
- Whitespace-Zeichen
 - Leerzeichen, Tabulator (horizontal, vertikal), neue Zeile, neue Seite