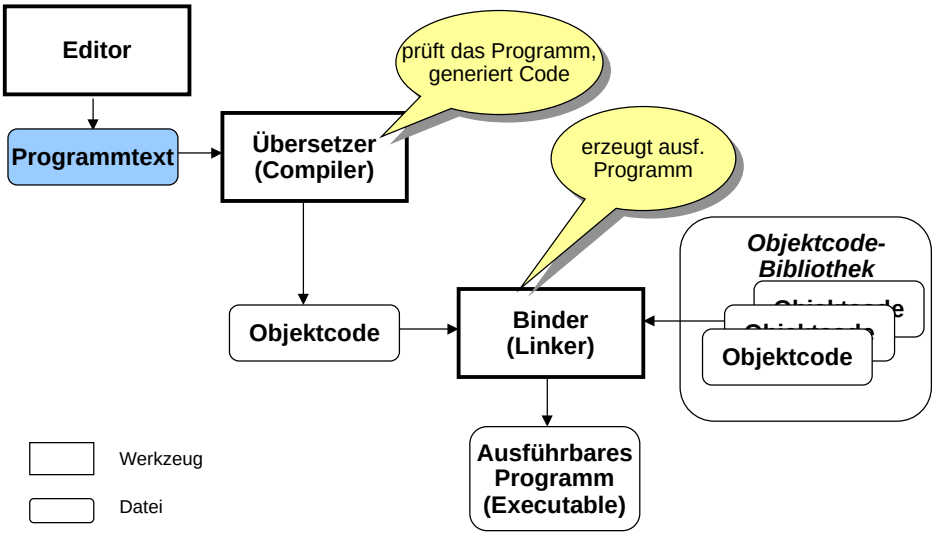
Infrastruktur für die Modula-3 Programmierung

- Hilfsmittel
- Editor und PM 3
- Das erste Modula-3 Programm
- Ein interaktives Programm
- Fehler und Warnungen
- Zusammenfassung

Hilfsmittel bei der Programmierung

- **Bei der Programmierung verwenden wir Hilfsmittel.**
  - Diese nennt man auch **Programmentwicklungswerkzeuge**.
  - Sie sind selbst wieder Programme.
- **Editor**
  - Erstellen des Programmtextes.
- **Übersetzer (Compiler)**
  - Dieser **prüft** den Programmtext auf Fehler und **übersetzt** den Programmtext in eine ausführbare Form.
- **Binder (Linker)**
  - Werden große Programme erstellt, bestehen diese nicht nur aus einer Programmdatei, sondern aus vielen.
  - Der Binder erzeugt daraus das **ausführbare** Programm.

Vom Programmtext zum ausf. Programm



Editor

- **Notwendig, um den Programmtext einzugeben**
- **Eigenschaften eines “guten” Programmeditors**
  - stellt Zeichen mit fester Breite dar
  - erzeugt reine Textdatei
  - Anzeige der Zeilennummer (z. B. Fehlersuche)
  - hebt syntaktische Elemente hervor
- **Geeignete Editoren**
  - Unix: xemacs, vi, emacs, xedit, ...
  - Windows NT: xemacs, vi, zur Not auch Wordpad, ...
  - SCHLECHT: Word, Framemaker, ...

PM 3

- **Polytechnique (Montreal) Modula-3**
  - aktuellste Modula-3 Freeware-Implementierung
  - Teil der Erstsemester-CD
  - basiert auf dem ursprünglichen DEC SRC Modula-3
- **Umfaßt den Compiler (Übersetzer)**
  - prüft die Syntax des Programmtexts
  - übersetzt in durch den Rechner ausführbaren Code
- **Linker (Binder) und vorgefertigte Bibliotheken**
  - für die Ein-/Ausgabe von Daten (z.B. das Modul IO)
  - um Zeichenketten zu manipulieren (z.B. das Modul Text)
- **Laufzeitumgebung**
  - stellt den Rahmen für ein ausführbares Programm
  - übernimmt die Speicherverwaltung

Erstes MODULA-3 Programm

- **Aufgabenstellung:**
  - Erstellen Sie ein Programm, das den folgenden Willkommensgruß auf dem Bildschirm anzeigt und dann beendet!

```
---------------------------------
Willkommen zum Studium in Aachen!
---------------------------------
```

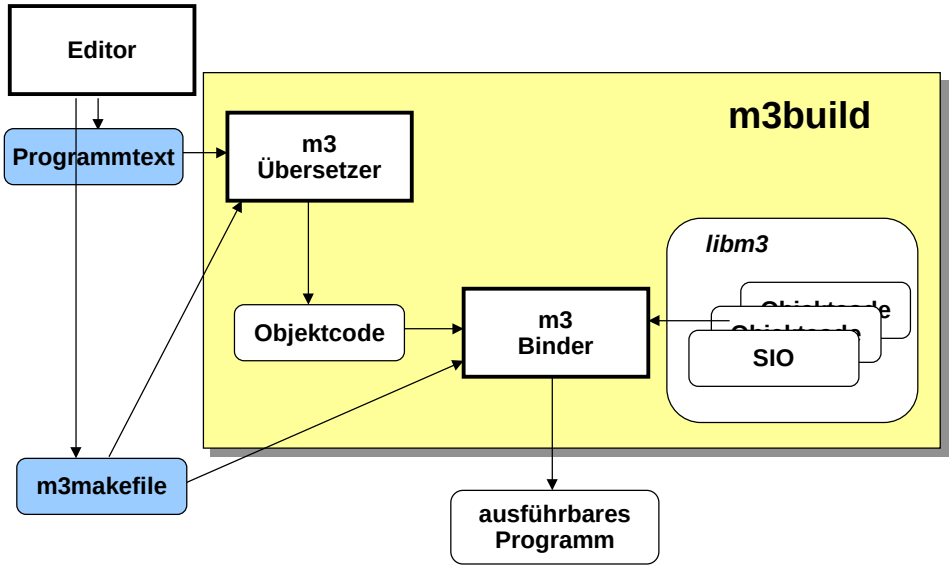
- **Lösungsidee:**
  - Da MODULA-3 Möglichkeiten anbietet, um Texte am Bildschirm auszugeben, können wir die Lösung zu dieser Aufgabenstellung direkt als Programm formulieren.
    - Modul SIO (Simple Input Output)
    - Stellt u.a. die Operationen
      - `SIO.PutText ()` und
      - `SIO.Nl()` zur Verfügung

Das erste M3-Programm

```
MODULE WillkommenInAachen EXPORTS Main;
(* Dieses Programm zeigt einen Willkommensgruss
   Autor           : Horst Lichter, RWTH Aachen
   Erstellt        : 16.08.98
   Letzte Aenderung: 20.08.98
*)

IMPORT SIO;
BEGIN
  SIO.Nl();
  SIO.PutText("--------------------------------");
  SIO.Nl();
  SIO.PutText("Willkommen zum Studium in Aachen!");
  SIO.Nl();
  SIO.PutText("--------------------------------");
  SIO.Nl();
END WillkommenInAachen.
```


Entwicklungswerkzeuge für Modula-3



m3build

- **Kommando auf der Ebene des Betriebssystems,**
  - um ein Modula3-Programm zu übersetzen und, wenn möglich,
  - daraus eine ausführbare Datei zu erzeugen
- **m3build erwartet,**
  - die Datei **m3makefile** im aktuellen Verzeichnis
  - Fehlt diese, so wird eine Fehlermeldung ausgegeben.
- **m3build**
  - verarbeitet den Inhalt von m3makefile
  - übersetzt das dort angegebene Programm
    - gibt eventuell Fehlermeldungen aus
  - erzeugt aus dem übersetzten Programm eine ausführbare Datei

Die Datei m3makefile

- **Liste der Teile**
  - um ein ausführbares Programm zu erstellen
  - gibt an, welche "Teile" benötigt werden
  - und wo sich diese befinden
- **Beispiel:**

```
% Makefile fuer Modula-3
% Programm WillkommenInAachen

import("libm3")
import("libSIO")

implementation("WillkommenInAachen")
program("WillkommenInAachen")
```

Standardmodul für Modula-3

Modul SIO

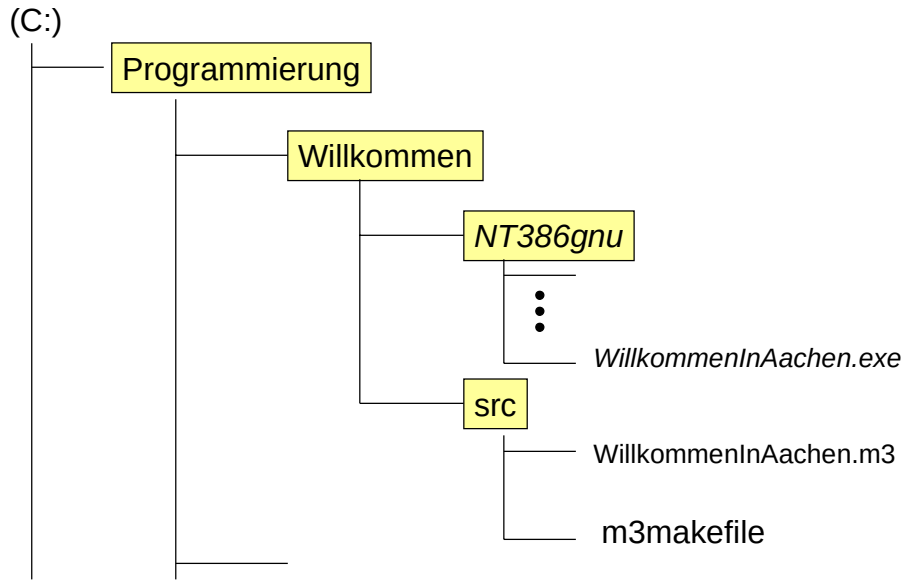
In der Datei <x>.m3 ist ein Modul implementiert

Verwende <x>.exe als Dateiname für das ausführbare Programm

Verzeichnisstruktur für m3build

- **src-Verzeichnis**
  - Hier muss die Datei, die den Programmtext enthält, abgelegt sein.
  - Diese muss die Endung "**.m3**" haben. Beispiel: Willkommen.m3
  - In diesem Verzeichnis sucht m3build die Datei m3makefile. Dementsprechend muss diese dort vorhanden sein.
  - Muss vom Programmierer angelegt werden.
  - Name unter Windows und UNIX: "**src**"
- **exe-Verzeichnis**
  - Wird automatisch von m3build angelegt (auf der selben Ebene wie das src-Verzeichnis!)
  - Nimmt alle Ergebnisse von m3build auf, insbesondere die ausführbare Datei. Diese hat z.B. die Endung "**.exe**"
  - Verzeichnisname unter Windows: "NT386gnu"
    Solaris: "SOLgnu"
    LINUX: "LINUXLIBC6"

Beispiel für Verzeichnisstruktur



Ein interaktives Programm

```
MODULE WillkommenInX EXPORTS Main;
(* Dieses Programm zeigt einen Willkommensgruss
   Autor           : Horst Lichter, RWTH Aachen
*)

IMPORT SIO;

VAR ort : TEXT;
BEGIN
  SIO.PutText ("Bitte Ort angeben: ");
  ort := SIO.GetLine();
  SIO.Nl();
  SIO.PutText("---------------------------------");
  SIO.Nl();
  SIO.PutText("Willkommen zum Studium in ");
  SIO.PutText(ort);
  SIO.Nl();
  SIO.PutText("---------------------------------");
  SIO.Nl();
END WillkommenInX.
```


Fehler

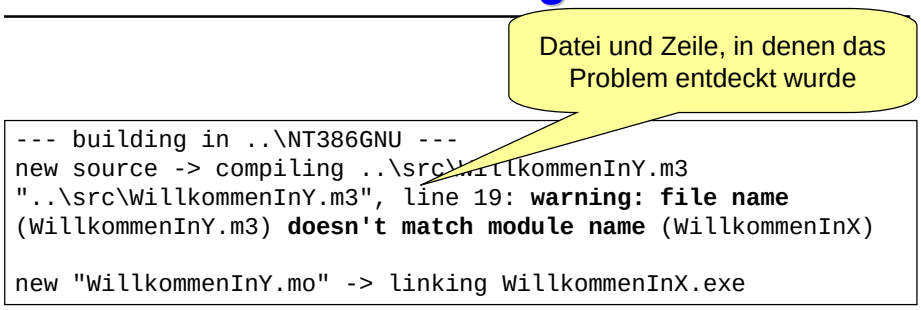
Datei und Zeile, in denen das Problem entdeckt wurde

```
--- building in ..\NT386GNU ---
new source -> compiling ..\src\WillkommenInX.m3
"..\src\WillkommenInX.m3", line 19: Initial module name
doesn't match final name (WillkommenInY)

compilation failed => not building program "WillkommenInX"
m3build: quake error:
```

- **Übersetzer gibt einen Fehler aus**
  - wenn eine Unstimmigkeit entdeckt wird
  - und die vollständige Übersetzung daher nicht mehr möglich ist
- **Ein guter Übersetzer lokalisiert Fehler sehr genau**
  - dennoch muss die angezeigte Stelle nicht Problemursache sein

Warnung




```
--- building in ..\NT386GNU ---
new source -> compiling ..\src\WillkommenInY.m3
"..\src\WillkommenInY.m3", line 19: warning: file name
(WillkommenInY.m3) doesn't match module name (WillkommenInX)

new "WillkommenInY.mo" -> linking WillkommenInX.exe
```

- **Der Compiler gibt eine Warnung aus**
  - wenn eine Unstimmigkeit entdeckt wird
  - die vollständige Übersetzung jedoch möglich ist
- **Warnungen**
  - zeigen an, daß etwas nicht stimmt und sollten geprüft werden
  - können aber oft ignoriert werden

Zusammenfassung

- **Editor und Compiler (PM 3) sind das Rüstzeug für die Programmierung**
- **Modula-3 erwartet eine feste Verzeichnisstruktur, beachten Sie diese!**
- **Für Ein-/Ausgaben verwenden wir das Modul SIO**
  - Import im Programm: `IMPORT SIO;`
  - im m3makefile mit: `import("libSIO")`
- **Achten Sie auf ordentlichen Programmierstil:**
  - Einrückungen
  - Wahl aussagekräftiger Bezeichner
  - Kommentare!!

