

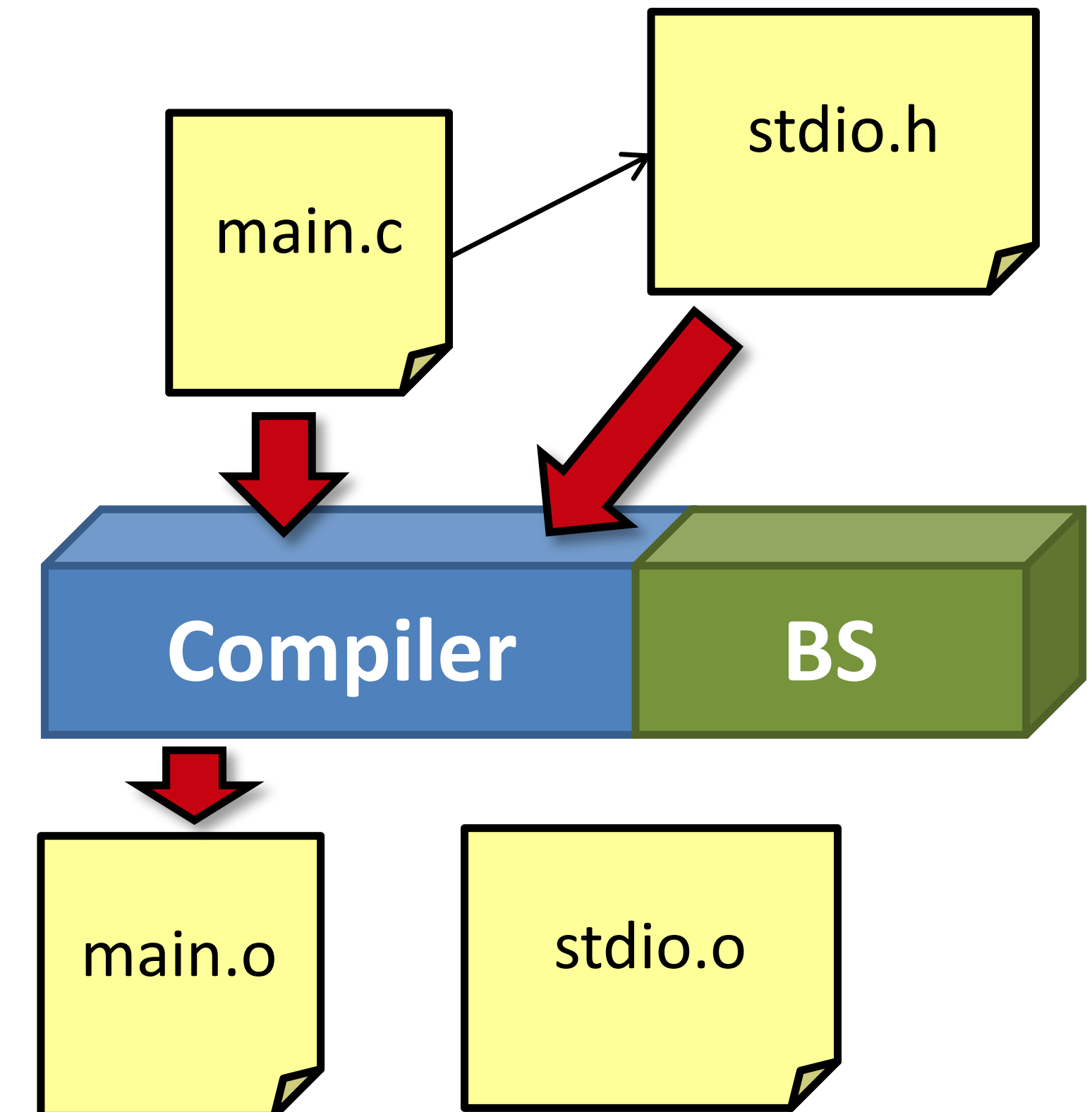
# Compiler und IDE

# Wie erzeugt man ein ablauffähiges Programm?

- C als Programmiersprache ist nur ein Text, der in ein Maschinen-ausführbares Format (Maschinencode) **übersetzt** werden muss (**Compiler**)
- Beispiel:
  - $x = y + 12$ ; muss nach Assembler übersetzt werden:
    - lade r1 mit Speicherstelle für y (ld [1234],r1)
    - Addiere 12 auf den Wert, den y hatte (addx r1,#12,r1)
    - Speichere Ergebnis in Speicherstelle für x (st r1,[5432])
  - Assembler-Programm ist wiederum nur eine Textdatei und muss in Maschinencode übersetzt werden (Assembler)
    - ld [1234],r1 -> 0100111010010100100100101
    - addx r1,#12,r1 -> 1001110100100100100111010
    - st r1,[5432] -> 0110010100111100101010

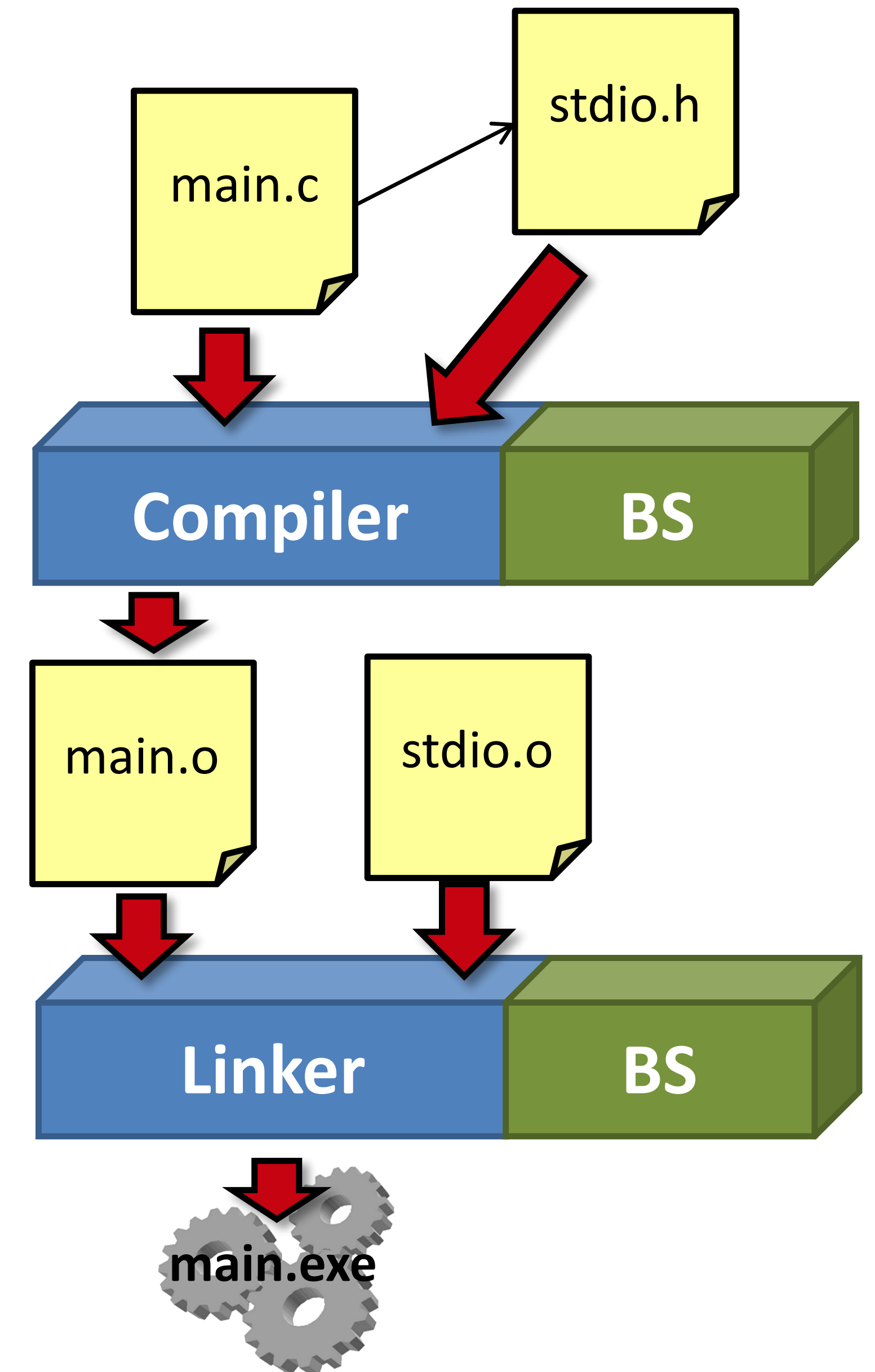
# Compiler

- Compiler hängen immer auch vom Betriebssystem ab, da das Betriebssystem Vorschriften macht, wie bestimmte Daten zu benutzen sind
- Compiler übersetzen das C-Programm in sogenannten Object-Code
- Object-Code enthält die zum Programm gehörigen Teile des Maschinen-Codes, ist aber nicht ausführbar
- Object-Code für vordefinierte Funktionen werden mit dem Compiler mitgeliefert, werden aber nicht vom Compiler übersetzt



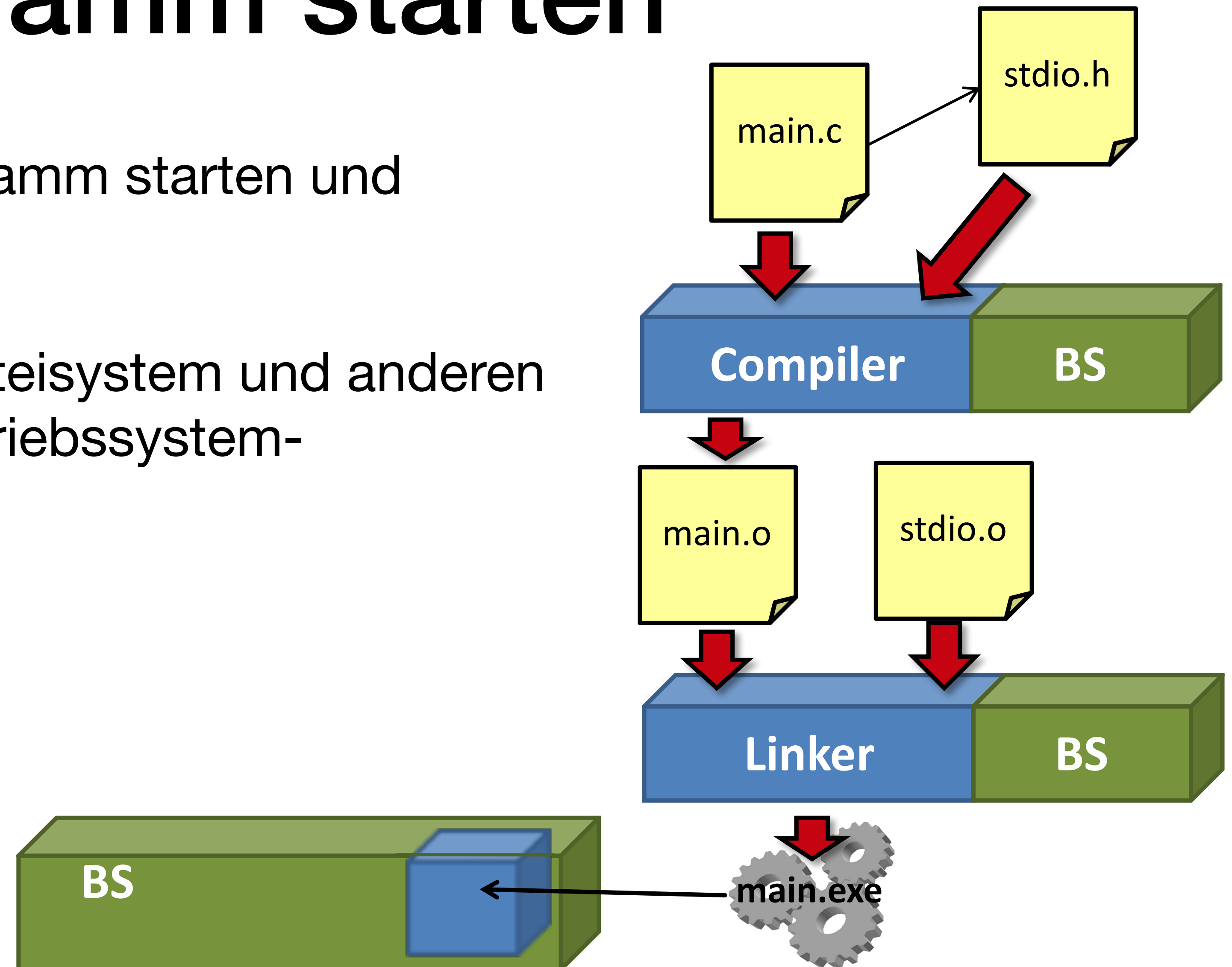
# Linker

- Object-Code aus verschiedenen Quellen muss zusammengefügt werden
- Referenzen müssen aufgelöst werden
- Betriebssystem-Code muss eingebaut werden, damit das Programm ausgeführt werden kann
- Ausführbares Programm main.exe entsteht



# Programm starten

- Betriebssystem muss Programm starten und stoppen können
- Interaktion mit Benutzer, Dateisystem und anderen Schnittstellen läuft über Betriebssystem-Funktionen

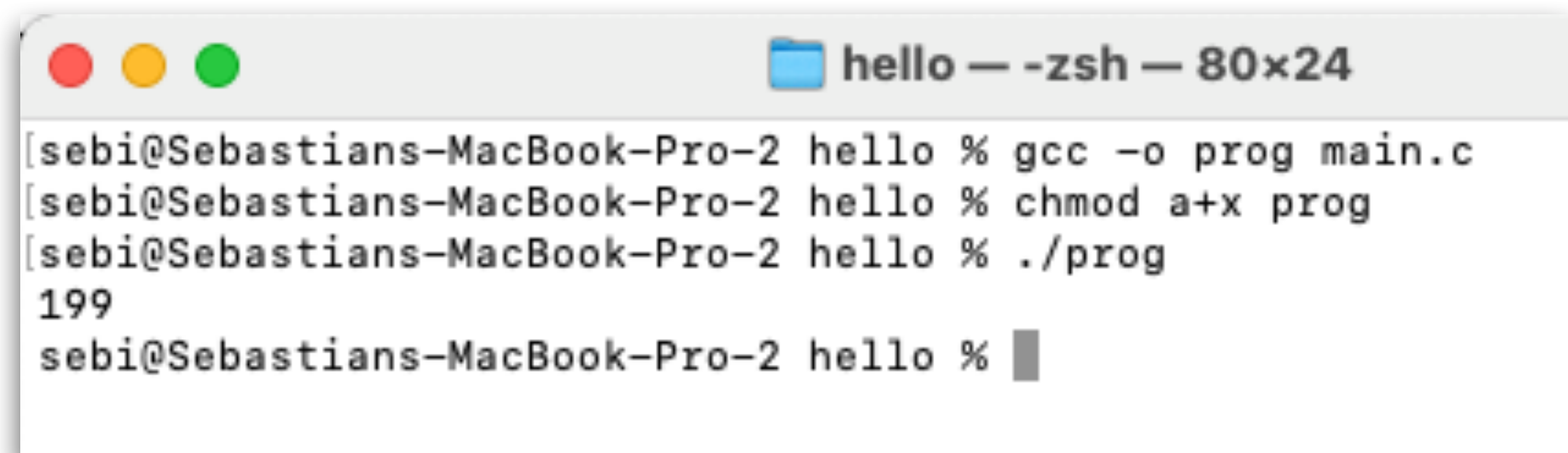


# Was muss der Programmierer wissen?

- Hängt davon ab, wie umfangreich das Programmier-Projekt ist
- Typischerweise werden IDEs (Integrated Development Environment) verwendet, die vieles für den Benutzer automatisch im Hintergrund machen
  - Stellt reichhaltige Editoren zur Verfügung
  - Übernimmt Übersetzung und Linken automatisch
  - Bietet benutzerfreundliche Schnittstellen zum Debugging
  - Ablauf des Programms durch Übergabe an Betriebssystem
- Bei komplexen Programmen muss der Programmierer Compiler und Linker manuell konfigurieren können

# Compiler

- z.B. gcc (C) oder g++ (C++)
  - gcc -o prog.exe main.c
- -o OUTPUTNAME CFILE
- Alles in einem: Compile + Link
- Nur Compile: -c

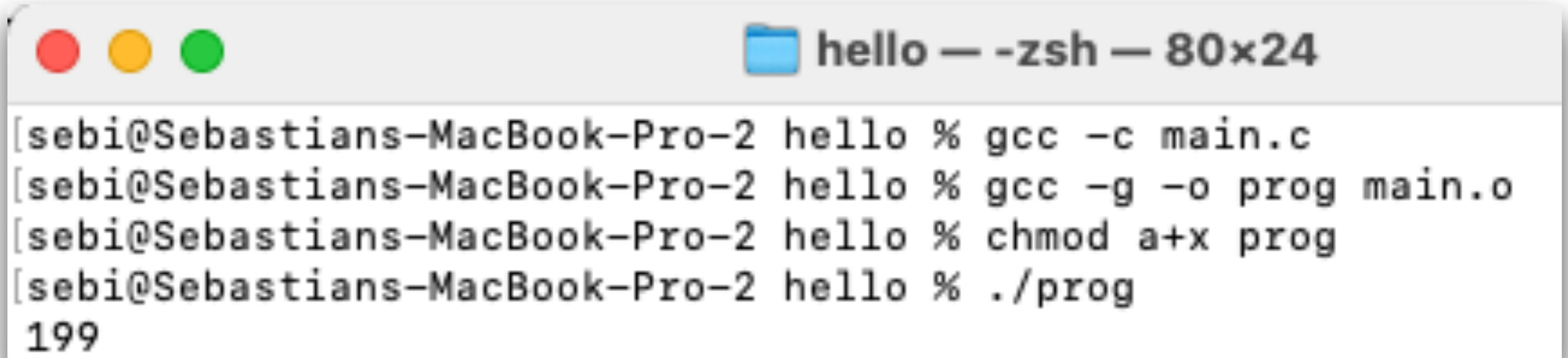


```
hello — -zsh — 80x24
[sebi@Sebastians-MacBook-Pro-2 hello % gcc -o prog main.c
[sebi@Sebastians-MacBook-Pro-2 hello % chmod a+x prog
[sebi@Sebastians-MacBook-Pro-2 hello % ./prog
199
sebi@Sebastians-MacBook-Pro-2 hello %
```



# Linker

- z.B. ld
  - `ld -o prog.exe main.o ...`
- oder gcc
  - `gcc -g -o prog.exe main.o ...`
- Alle Dateien zum Linken angeben

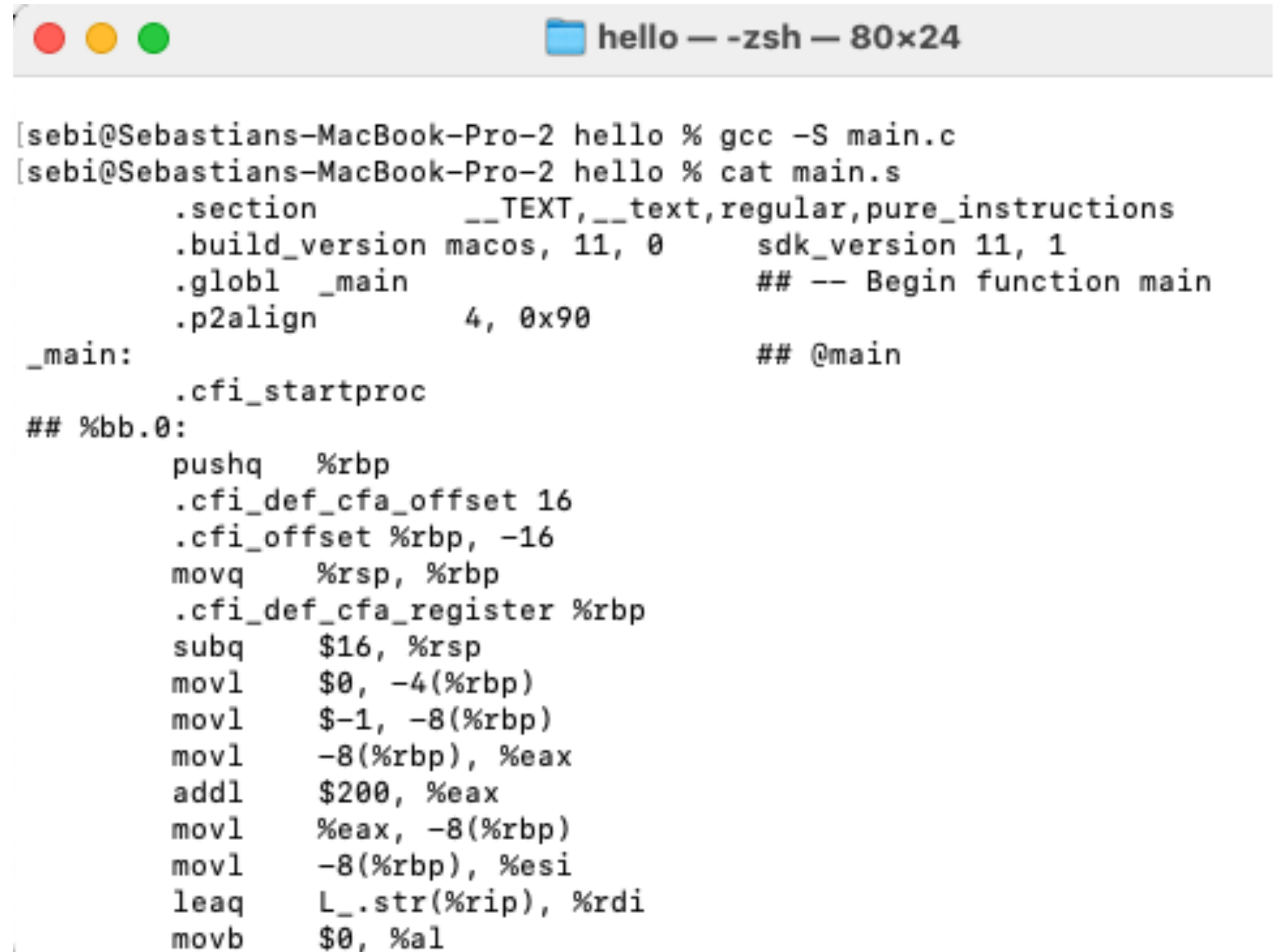


```
hello — -zsh — 80x24
[sebi@Sebastians-MacBook-Pro-2 hello % gcc -c main.c
[sebi@Sebastians-MacBook-Pro-2 hello % gcc -g -o prog main.o
[sebi@Sebastians-MacBook-Pro-2 hello % chmod a+x prog
[sebi@Sebastians-MacBook-Pro-2 hello % ./prog
199
```



# Assembler

- gcc -S
- Produziert Assembler Code
- Vorher noch Pre-processing nötig, damit die Header hinzugefügt werden (z.B. mit cpp)



```
hello — -zsh — 80x24

[sebi@Sebastians-MacBook-Pro-2 hello % gcc -S main.c
[sebi@Sebastians-MacBook-Pro-2 hello % cat main.s
        .section      __TEXT,__text,regular,pure_instructions
        .build_version macos, 11, 0      sdk_version 11, 1
        .globl _main                      ## -- Begin function main
        .p2align      4, 0x90

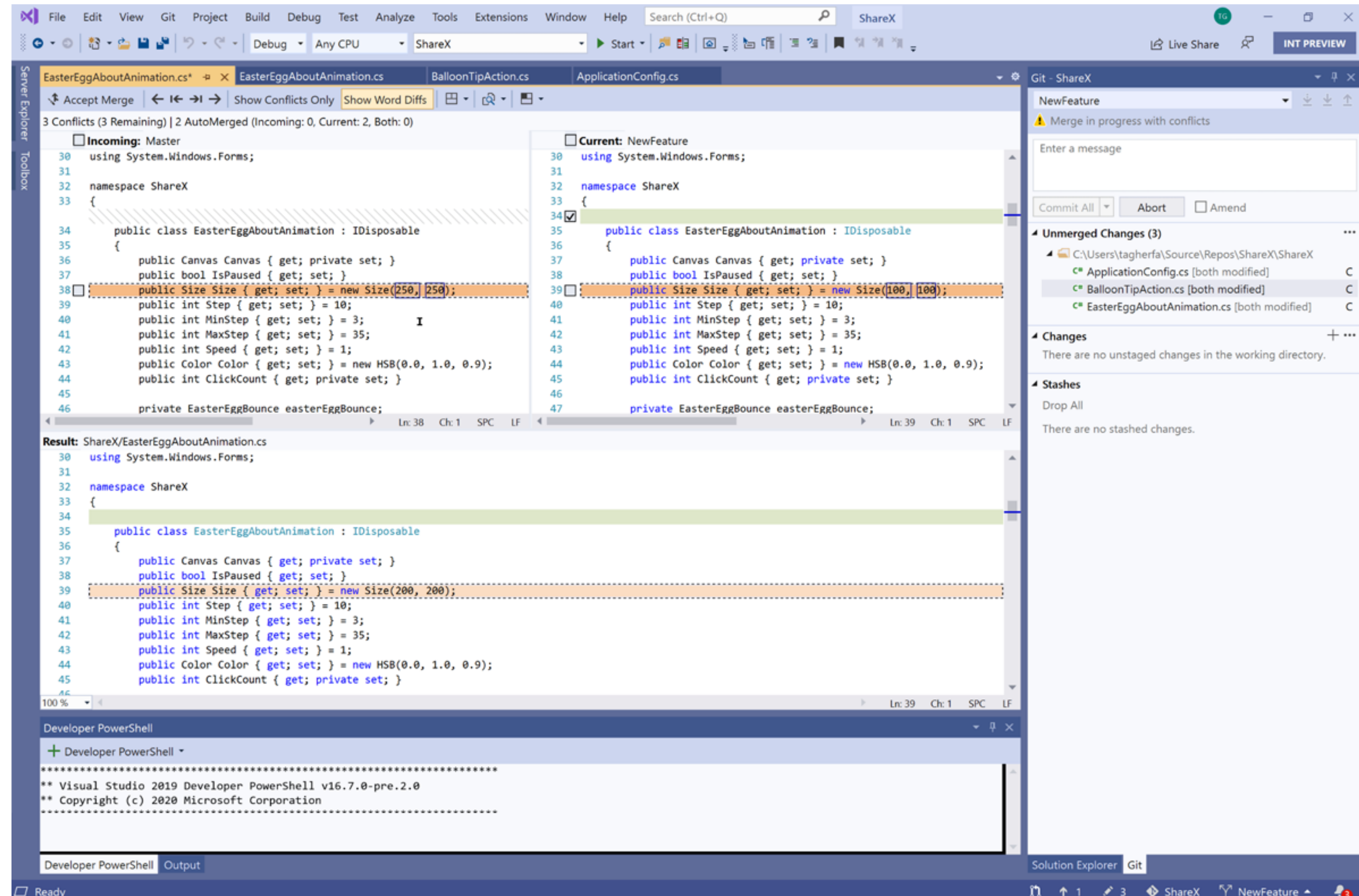
_main:                                     ## @main
        .cfi_startproc
## %bb.0:
        pushq    %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset %rbp, -16
        movq     %rsp, %rbp
        .cfi_def_cfa_register %rbp
        subq     $16, %rsp
        movl     $0, -4(%rbp)
        movl     $-1, -8(%rbp)
        movl     -8(%rbp), %eax
        addl     $200, %eax
        movl     %eax, -8(%rbp)
        movl     -8(%rbp), %esi
        leaq     L_.str(%rip), %rdi
        movb     $0, %al
```

# IDE

- Integrated Development Environment
- Windows: MS Visual Studio, Visual Studio Code, Eclipse, ...
- Linux: Visual Studio Code, KDevelop, Eclipse, ...
- Mac OS: Xcode, Visual Studio Code, ...
- Online IDEs: replit, ...
- Oder einen Texteditor und den Compiler, Linker per Kommandozeile (Debugger: gdb)

# Visual Studio

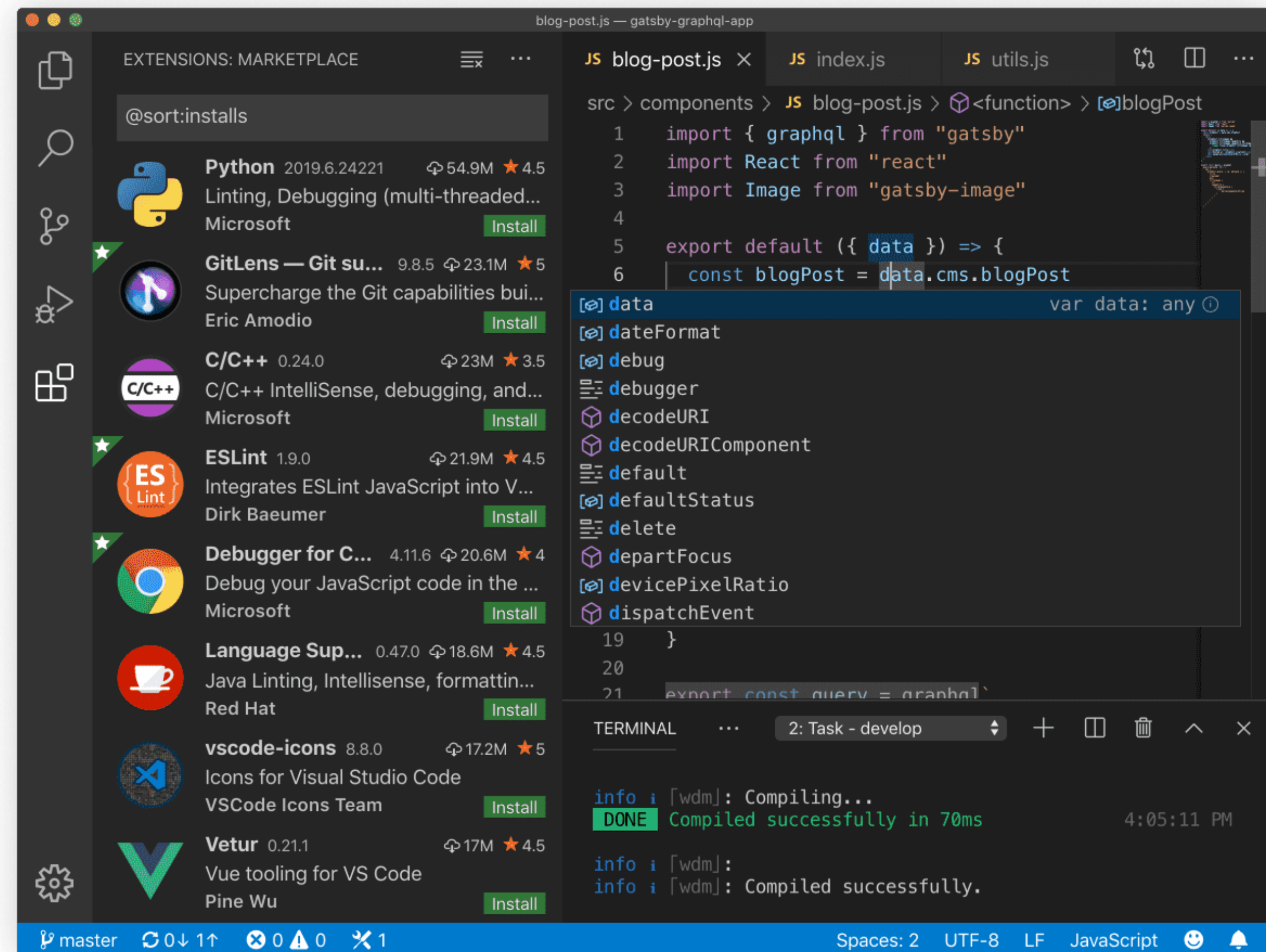
- Von Microsoft: <https://visualstudio.microsoft.com>
- Sehr Umfangreich;  
viele Funktionen; viele  
Programmiersprachen
- Auch für große  
Projekte geeignet





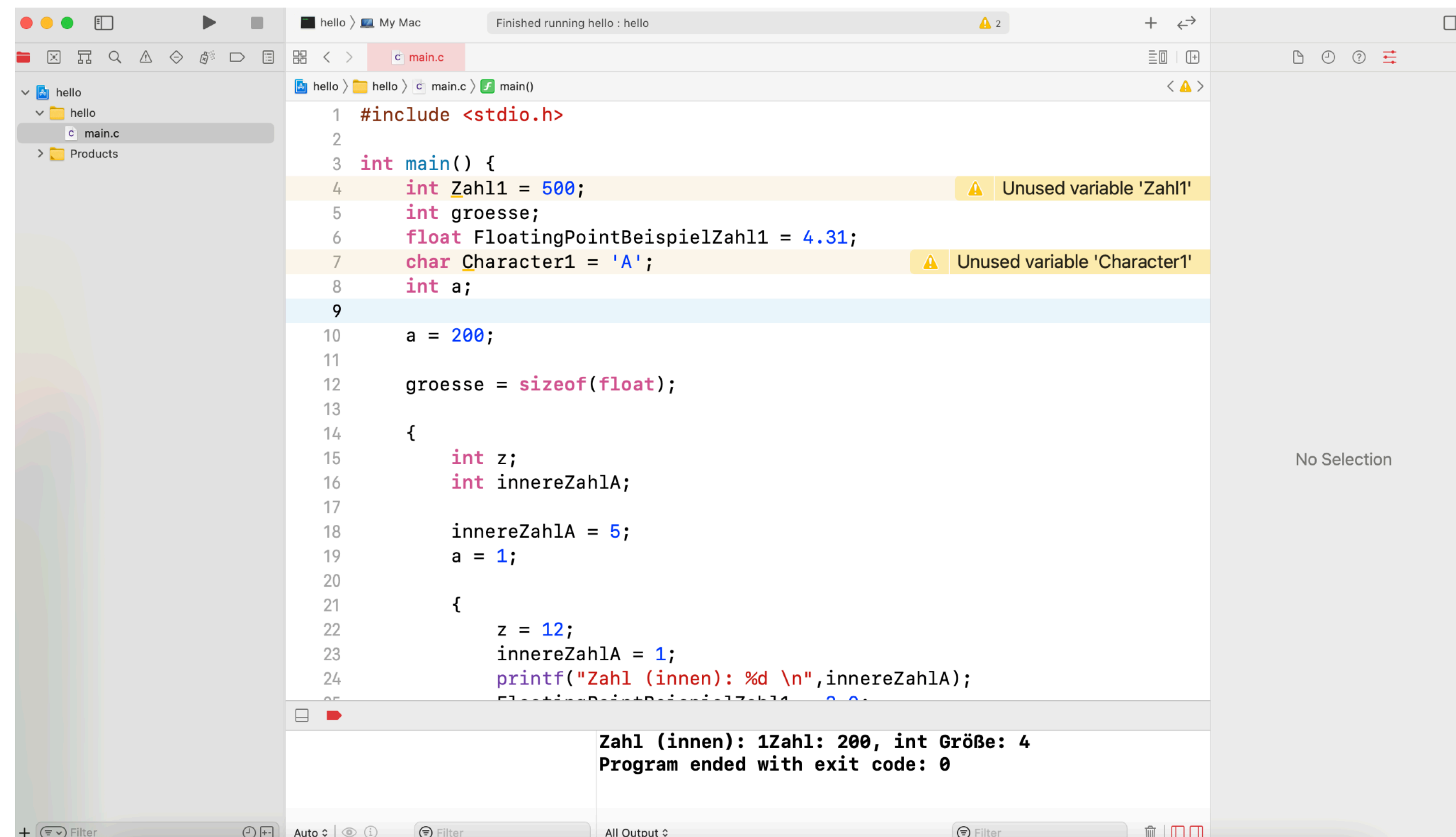
# Visual Studio Code

- Auch von Microsoft: <https://code.visualstudio.com>
- Editor mit vielen Erweiterungsmöglichkeiten
- Schnell, schlank, erweiterbar
- Anleitung MS C++: <https://code.visualstudio.com/docs/cpp/config-msvc>
- Anleitung GCC: <https://code.visualstudio.com/docs/cpp/config-mingw>
- Anleitung MacOS: <https://code.visualstudio.com/docs/cpp/config-clang-mac>



# Xcode

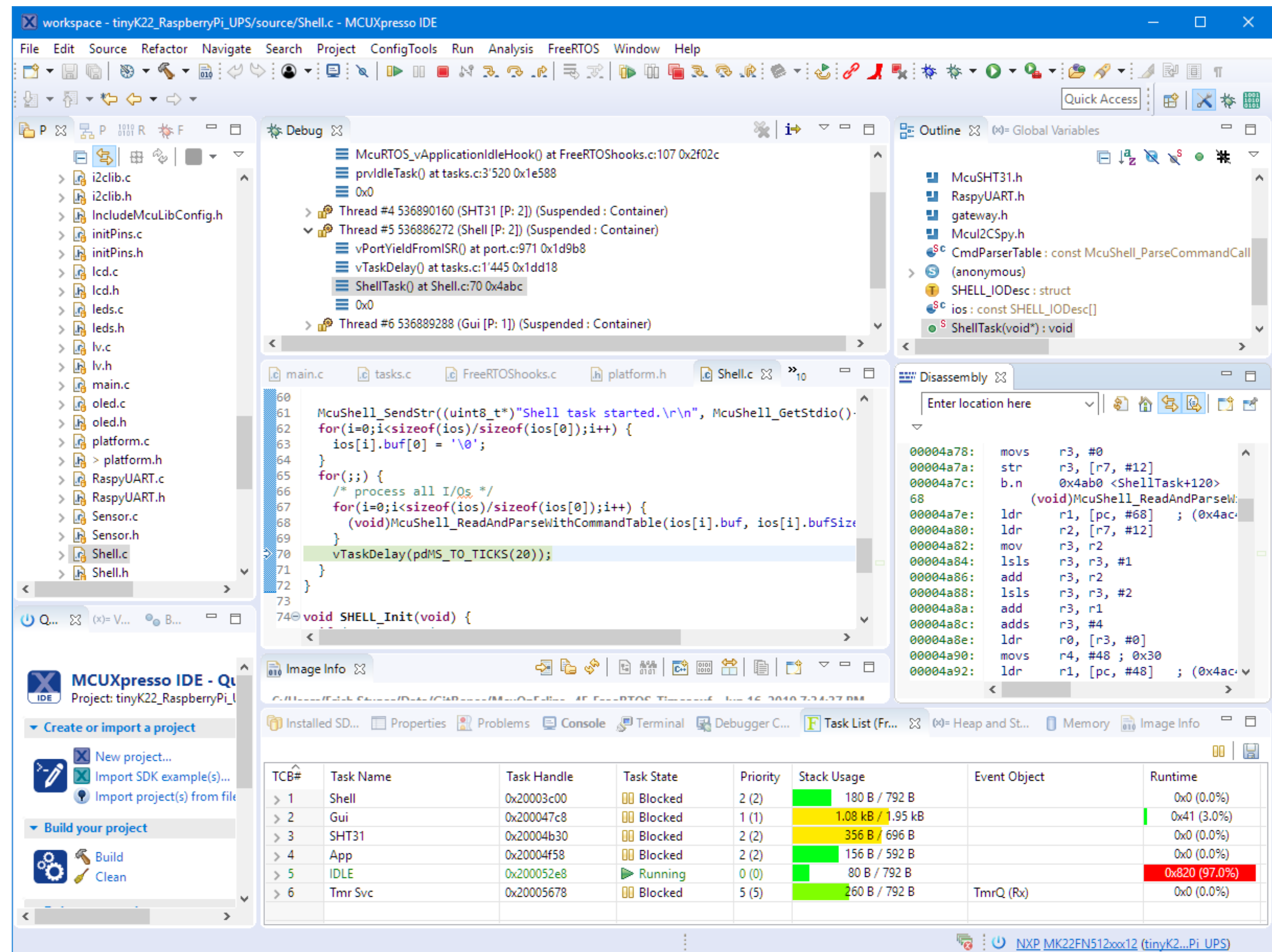
- Von Apple: <https://developer.apple.com/xcode/>
- Umfangreiche, einfache IDE
- Gut in MacOS integriert





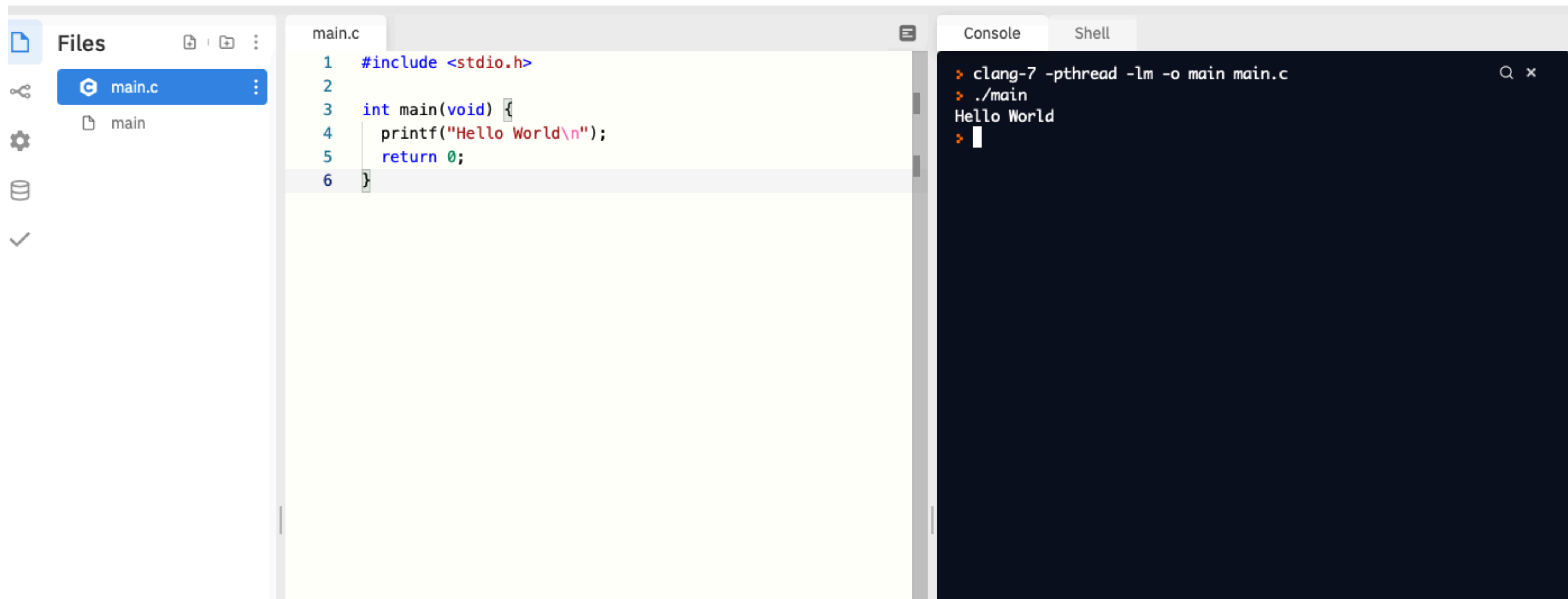
# Eclipse

- OpenSource: <https://www.eclipse.org/ide/>
- Umfangreich und viele Funktionen
- Unterstützt viele Sprachen



# replit

- Online IDE: <https://replit.com>





# Empfehlungen

- Einfach ausprobieren!
- Jeder hat andere Vorlieben und kommt unterschiedlich gut mit den IDEs zurecht