Einführung in die Programmierung mit C++

Übungsblatt 1

Organisatorisches
Entwicklungsumgebung
Hello World

Sebastian Christodoulou Alexander Fleming Uwe Naumann

Informatik 12:
Software and Tools for Computational Engineering (STCE)

RWTH Aachen

- Globalübung
  - Dienstag, 14:30-16:00 Uhr
  - AH III
  - Präsentation der Musterlösung zu den Übungsaufgaben der Vorwoche
  - Vorstellung der kommenden Übung
- Übungsaufgaben
  - Veröffentlichung Dienstags auf RWTHmoodle
  - Abgabe bis Dienstag, 14:30 Uhr auf RWTHmoodle
  - Bewertung durch Tutoren

- Prompt: Blickender Cursor nach der Ausgabe `stce@stce-vbox: $`
- Username: `stce`
- Hostname: `stce-vbox`
- Hilfe für Kommandos:
  - `man g++` zeigt die Hilfeseite von g++ an
  - `g++ --help` zeigt die Kurzversion der Hilfeseiten von g++ an

- pwd zeigt den Pfad des aktuellen Verzeichnisses an.
- `ls` (list) listet Inhalt des aktuellen Verzeichnisses auf
- `ls -la` zeigt Zusatzinformationen über die Dateien an
- `ls -t` zeigt sortiert die Ausgabe angefangen mit den jüngsten Dateien
- Das Sonderverzeichniss "." ist Synonym für das aktuelles Verzeichnis
- Das Sonderverzeichniss ".." ist Synonym für das übergeordnete Verzeichnis (..)
- Das Sonderverzeichniss "~" ist Synonym für das Heimatverzeichnis des aktuellen Users

Software and Tools for Computational Engineering

RWTH AACHEN UNIVERSITY

Übung:

- 'cd /home' wechselt das Verzeichnis
- 'cd /home/stce' wechselt in das Heimatverzeichnis des User stce
- 'cd' wechselt in das Heimatverzeichnis des aktuellen Users
- 'cd ~' wechselt in das Heimatverzeichnis des aktuellen Users
- 'mkdir test' erstellt ein Verzeichnis
- 'rmdir test' Löscht das Verzeichnis test wenn dieses leer ist.
- 'rm -rf test' Löscht das Verzeichnis test rekursiv unabhängig davon ob es leer ist oder nicht.
- 'ls' Dateien des aktuellen Verzeichnises anzeigen.
- 'ls -la test' Dateien in detailierter Ansicht anzeigen.

- Standard Texteditor von Unix ist der `vi`
- Zahlreiche Einführungsseiten finden sich im Internet
- Eine Erweiterung ist zum Beispiel der `vim` (vi improved)
- 2 Modi: Kommandomodus und Einfügemodus
- Mit 'i' wechselt man in den Einfügemodus
- Mit `ctrl-c` oder `escape` wechselt in den Kommandomodus zurück
- Es existieren zahlreiche Kommandos zur Editierung:
- `x` löscht Zeichen unter dem Cursor
- `0` springt an den Anfang der aktuellen Zeile
- `dd` löscht sie Zeile in der der Cursor steht

- `23dd` löscht 23 Zeilen ab Position an der Cursor steht
- `dw` löscht Wort nach Cursor
- `:w` schreibt den aktuellen Buffer
- `:q` Beendet den vi mit Prüfung auf Änderungen
- `:q!` Beendet den vi ohne Prüfung
- ... mehr findet sich in der Literatur oder im Netz

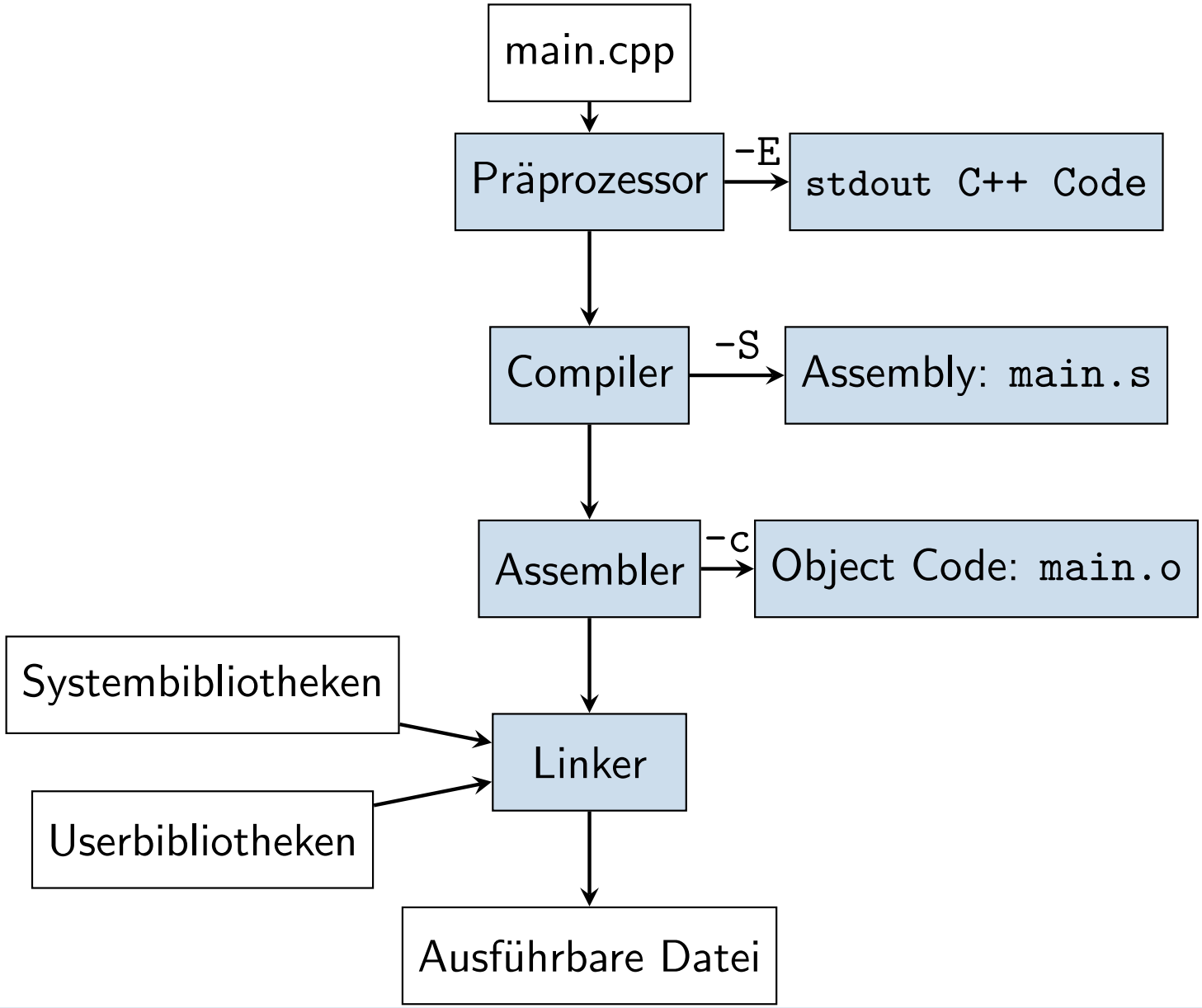
- `cp file1.ext file2.ext` kopiert file1.ext nach file2.ext
- `cp file1.ext /home/ab123456` kopiert file1.ext in das Heimatverzeichnis
- `cp -r folderxy /home/ab123456` kopiert Verzeichnis rekursiv in das Heimatverzeichnis
- `mv folderxy /home/ab123456` verschiebt Ordner folderxy in das Heimatverzeichnis
- `rm file1.ext` löscht die Datei file1.ext
- `rm -rf folderxy` löscht Ordner folderxy
- `rm a*` löscht alle Dateien die mit 'a' beginnen

- Sublime Text
  `subl`
- NeoVIM
  `nvim`
- Visual Studio Code (VSCode)
  `code`

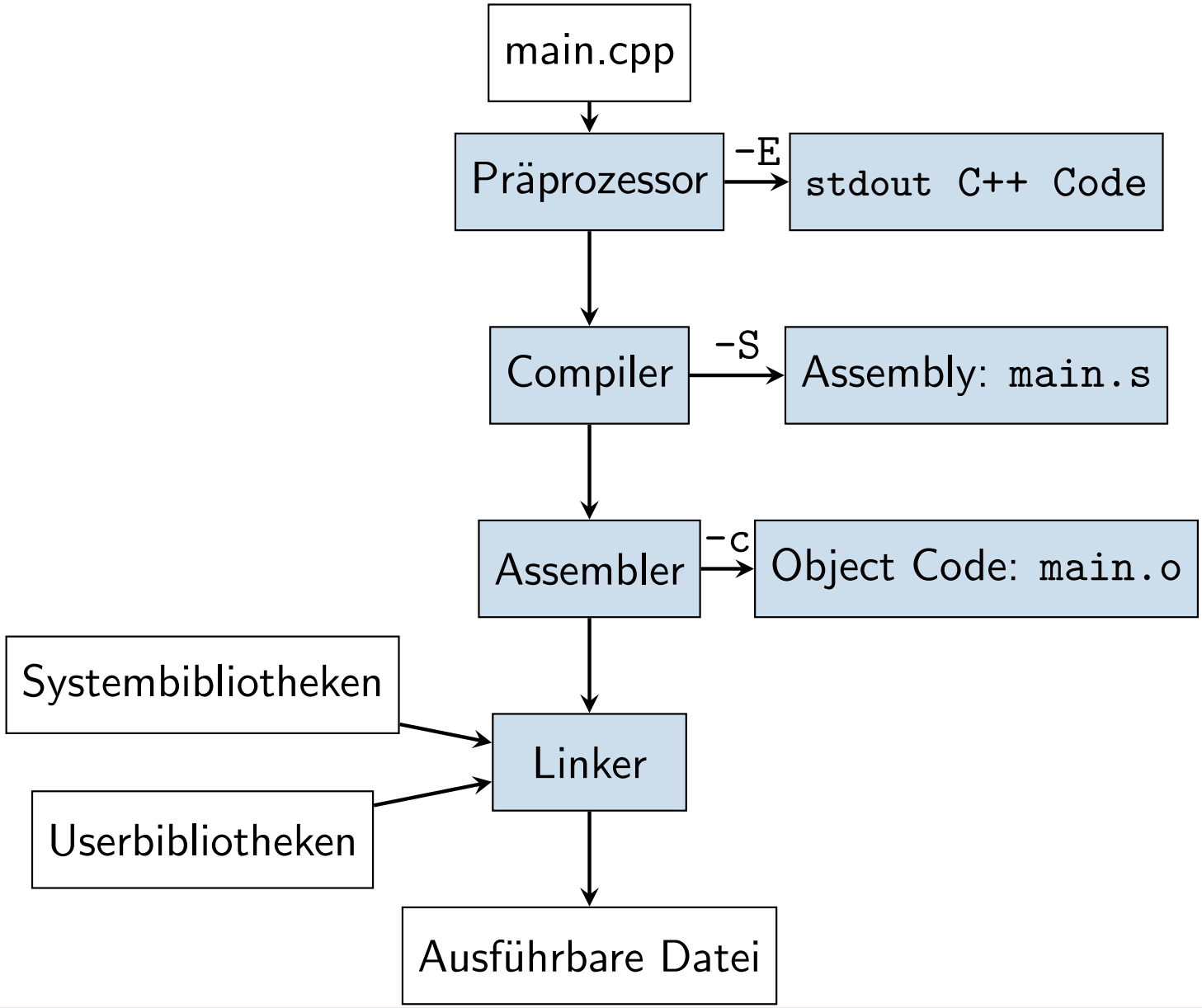
- `g++ -c main.cpp` erstellt aus Datei main.cc die Objektdatei main.o
- `g++ -o main.exe main.cpp` erzeugt die ausführbare Datei `main.exe`
- `g++ -o main.exe main.o` erzeugt die ausführbare Datei `main.exe`
- `g++ -o main.exe main.o`
- `-E` stoppt nach Präprozessorlauf und schreibt Ergebnis auf stdout
- `g++ -S main.cpp` erzeugt `main.s` in der Assemblercode steht
- `g++ -M main.cpp` zeigt die Abhängigkeiten der Datei main.cpp
- `-Wall` zeigt alle möglichen Warnungen an
- `-Werror` Warnungen werden als Fehler betrachtet
- `-ansi` kompiliert nach Ansi Standard

- `-l<library name>`
- `-L<path to library search path>`
- `-I<include path>`
- `-static`

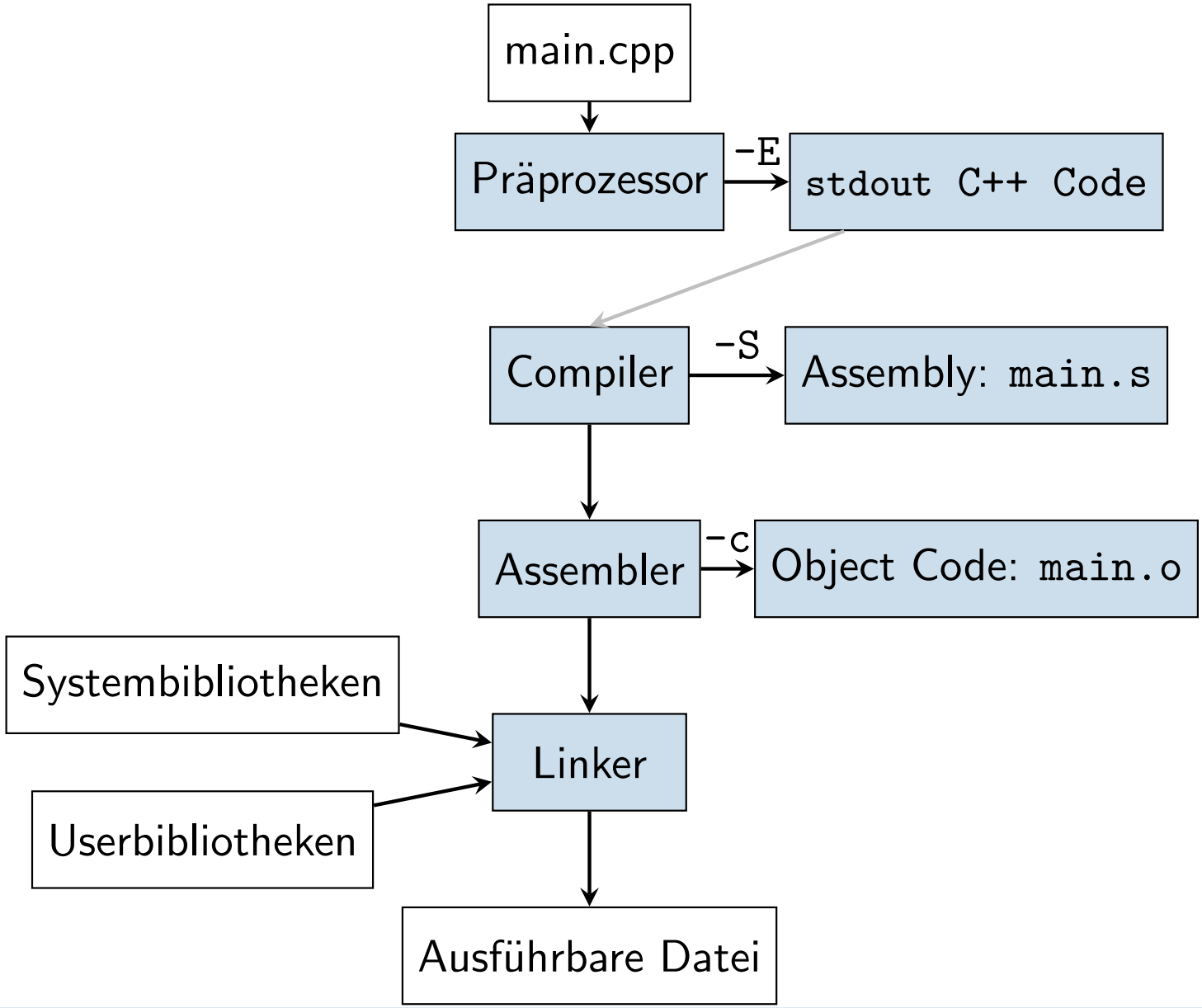
Einfacher g++ Flowchart



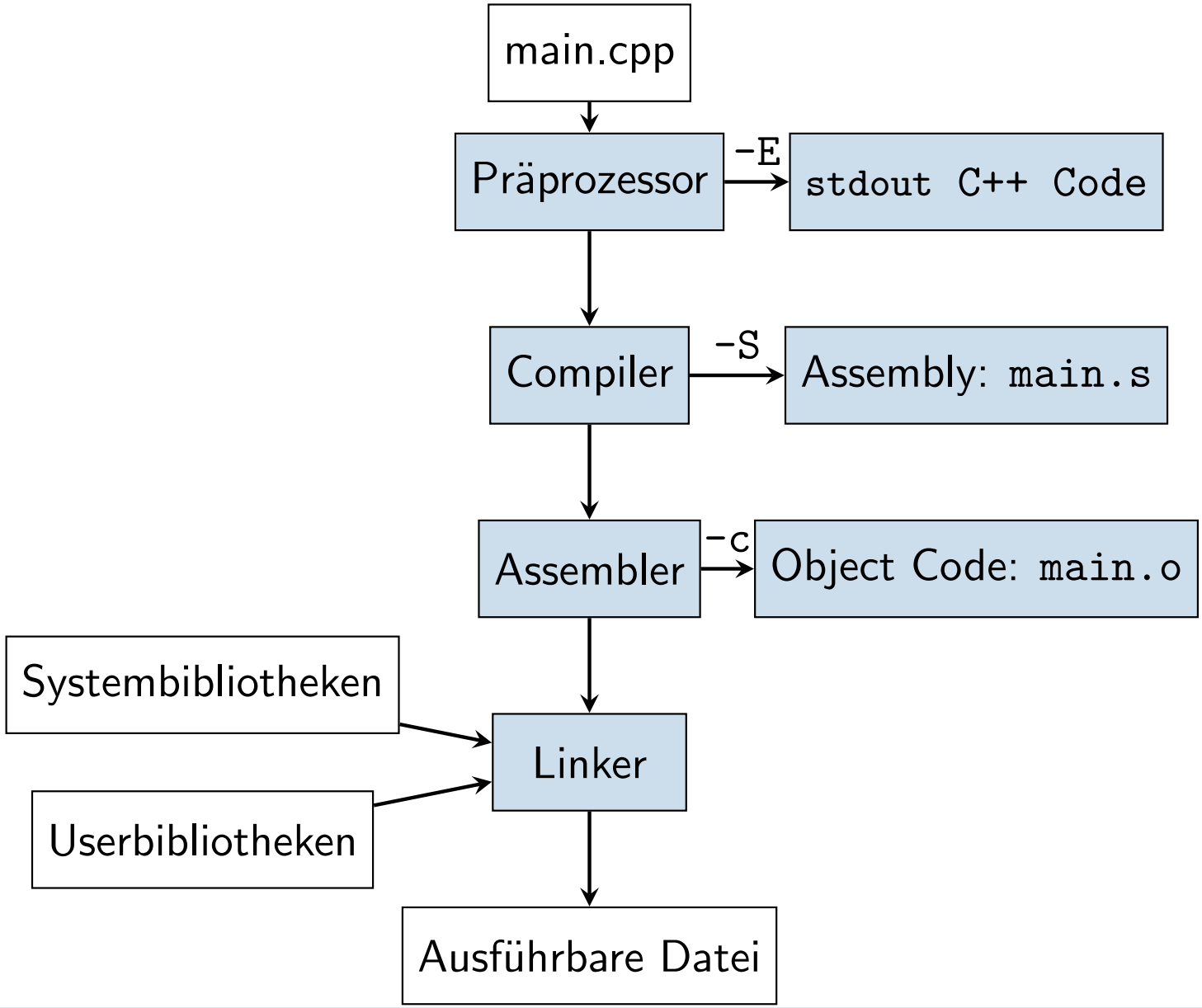
Einfacher g++ Flowchart



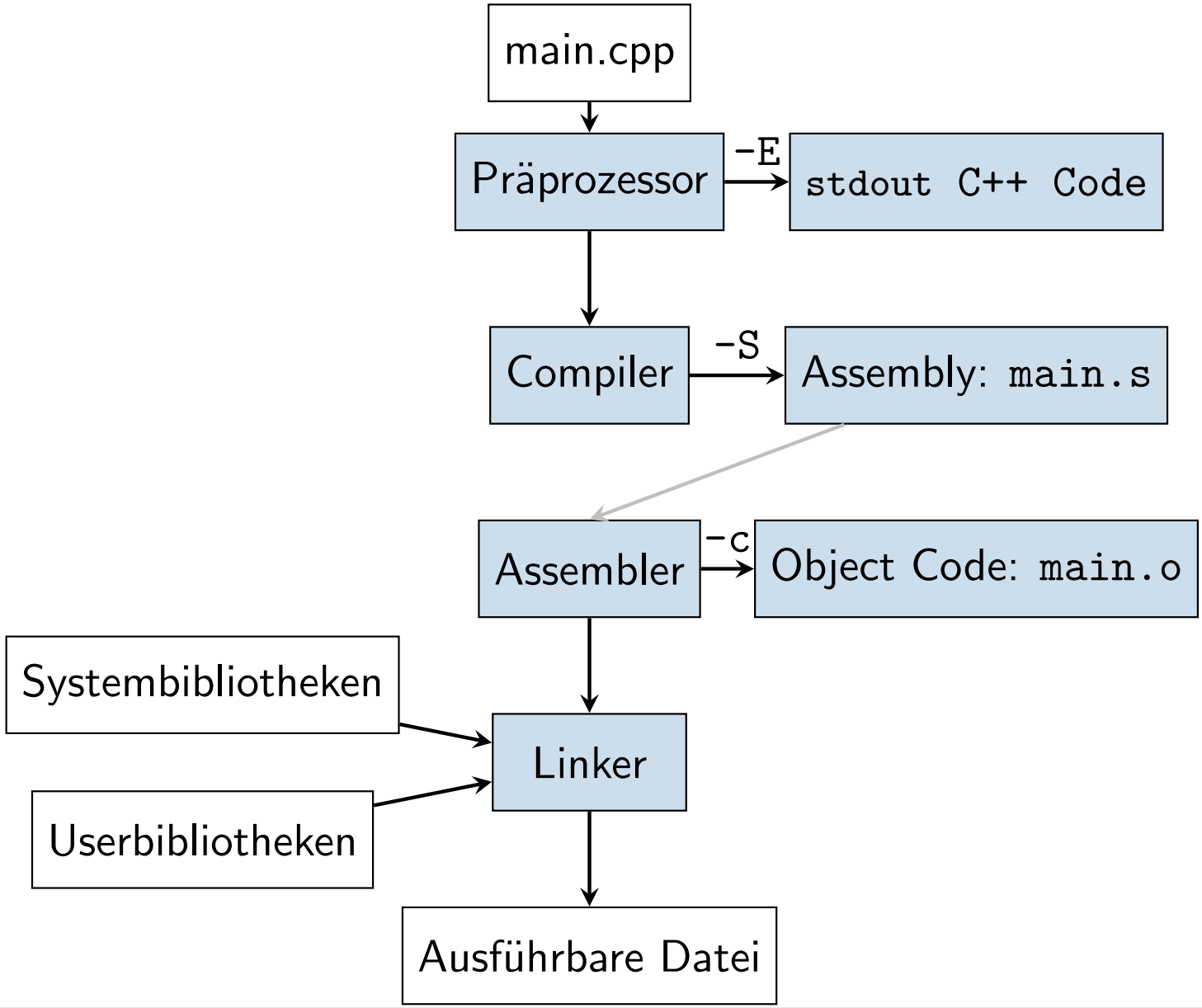
Einfacher g++ Flowchart



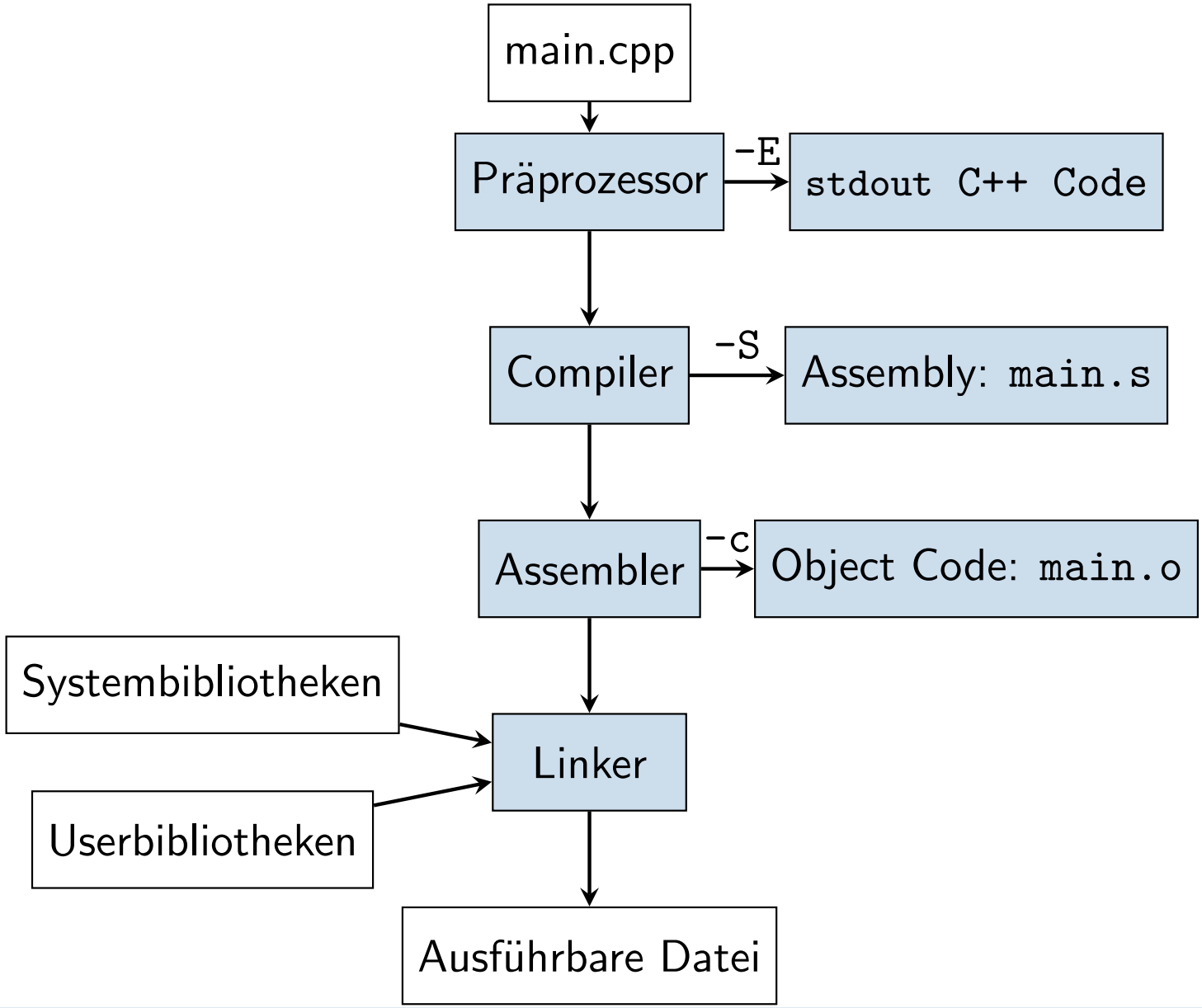
Einfacher g++ Flowchart



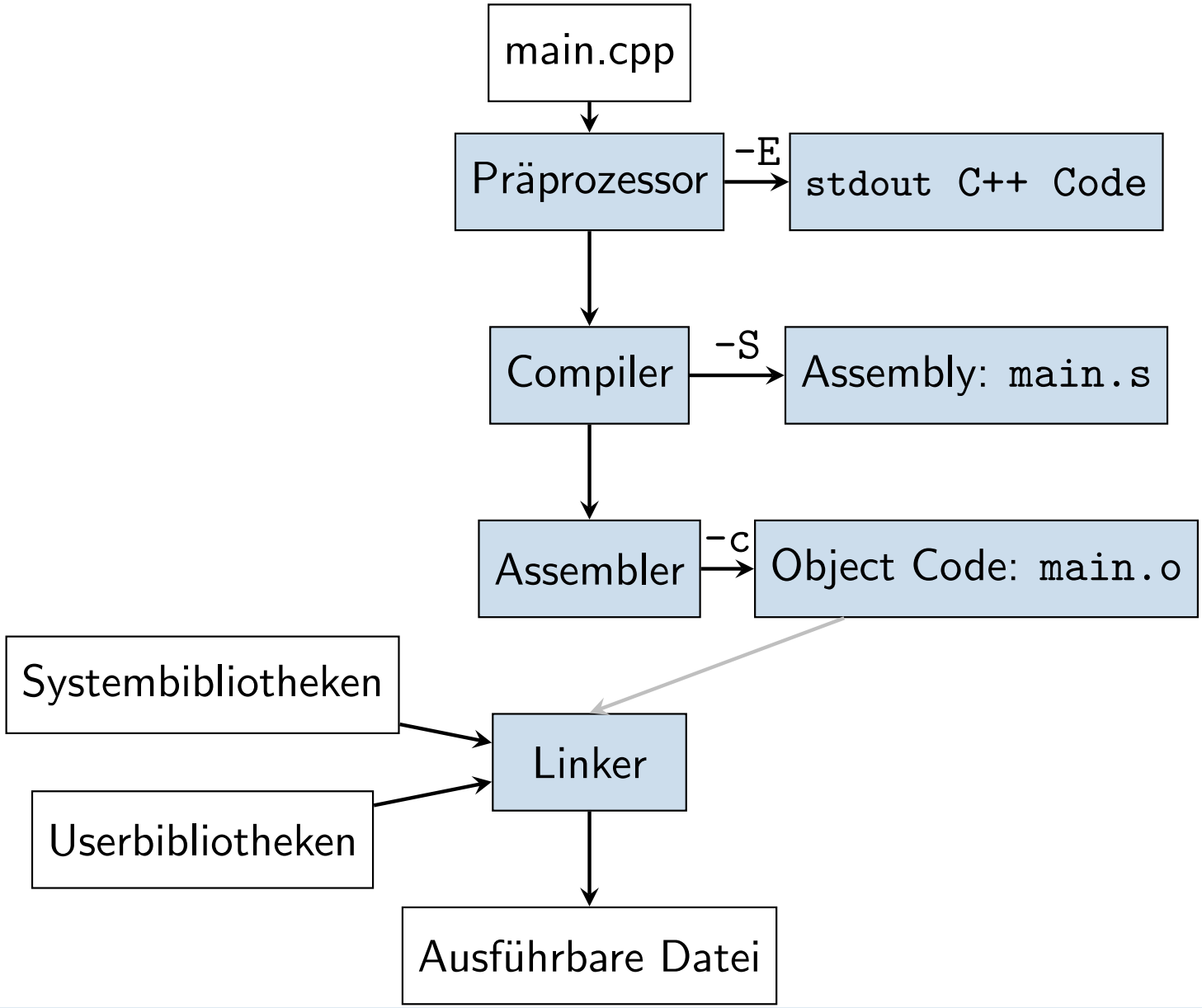
Einfacher g++ Flowchart



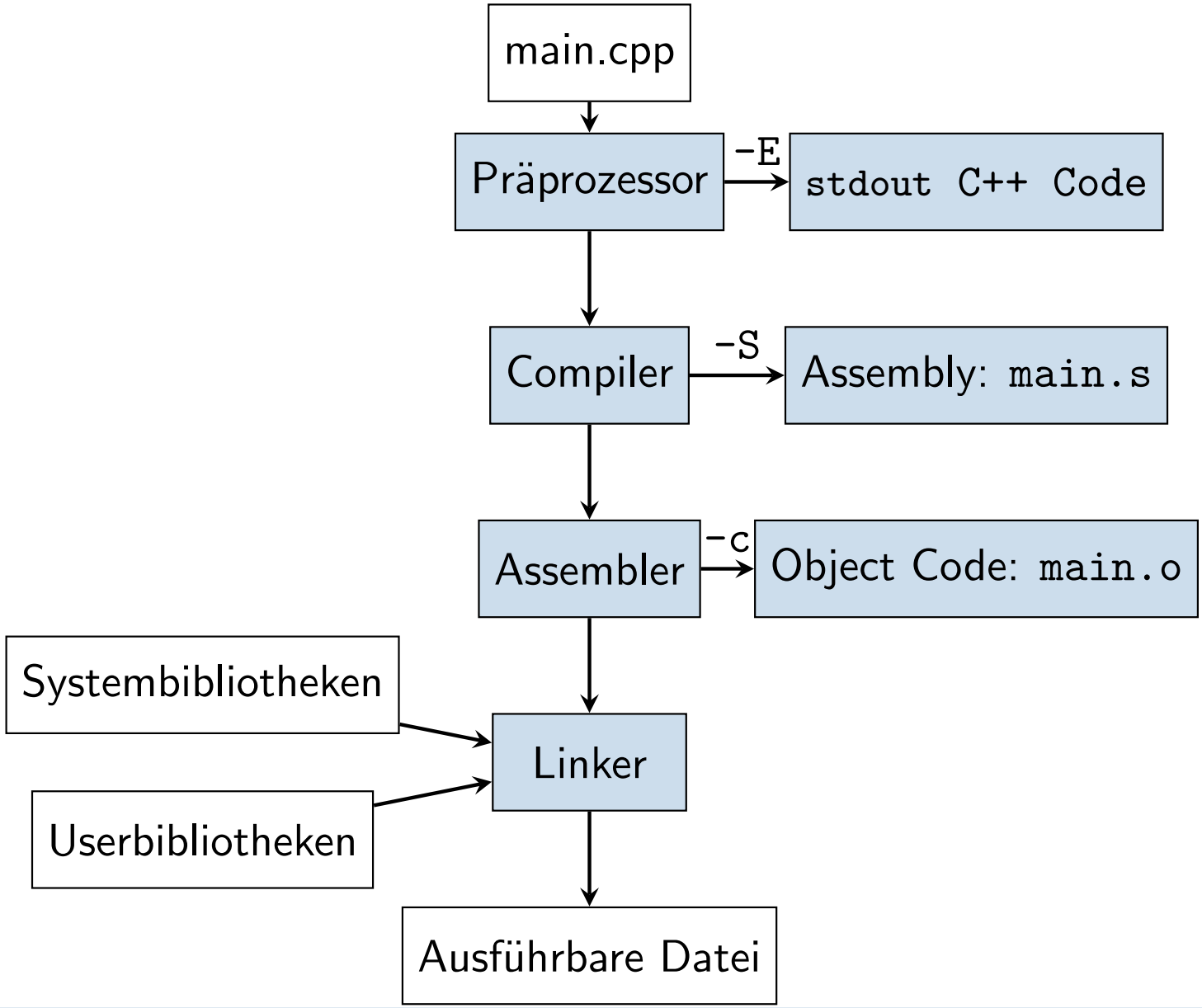
Einfacher g++ Flowchart



Einfacher g++ Flowchart



Einfacher g++ Flowchart



```
#include <iostream>

int main() {
  std::cout << "Hello World!" << std::endl;
}
```

- Dateiname sei main.cpp
- Kompilieren Sie das Programm mit `g++ main.cpp`
- Wird kein Name per `-o` angegeben wird die Datei `a.out` erzeugt
- Starten Sie das Programm mit `./a.out`

1. Erstellen Sie die Datei `hello.cpp` mit folgendem Inhalt:

```
#include <iostream>

int main() {
  std::cout << "Hello World!" << std::endl;
}
```

2. Versehen Sie die einzelnen Zeilen mit erklärenden Kommentaren.
3. Erläutern Sie kurz, was die folgenden Compileraufrufe bewirken.

```
g++ -E hello.cpp
g++ -c hello.cpp
g++ hello.o
g++ -o hello hello.cpp
g++ -Wall -Werror hello.cpp
```

Die Kurzhilfe des g++ Compilers erhalten Sie mit dem Konsolen-Kommando
`g++ --help | less`
und die Dokumentation mit
`man g++`
Die Angabe `| less` sorgt für eine seitenweise Ausgabe auf der Konsole.

Schreiben Sie Ihre Erläuterungen in eine Textdatei namens `README`.

4. Erweitern Sie die Textdatei mit dem Namen `README`: Erläutern Sie die Schritte, die nötig sind um eine ausführbare Datei zu erhalten und wie diese dann aufgerufen werden muss.

Abgabe:

- hello.cpp
- README (README.txt ist auch ok)