



# Working with multiple files

Stefan D. Bruda

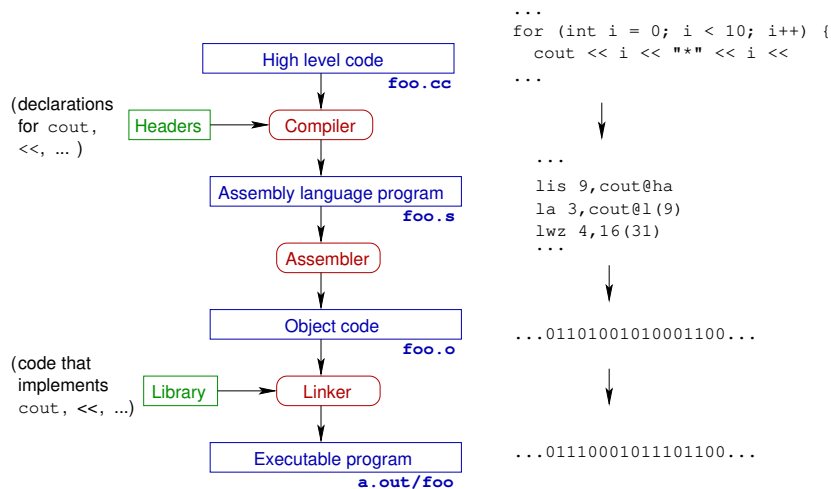
CS 464/564, Fall 2023

- Often we want to split our program into multiple files (or **modules**)
- Advantages: encapsulation, reusability, size
  - Compilation time also reduced
- A **module** consists of two parts:
  - the **header file**, where all the declarations available outside the module go (e.g., `tcp-util.h`)
  - the **C/C++ code** which implements the things declared in the header (e.g., `tcp-util.cc`)
- Another module (say `main.cc`) that wants to use `tcp-util.cc` will use the directive
 

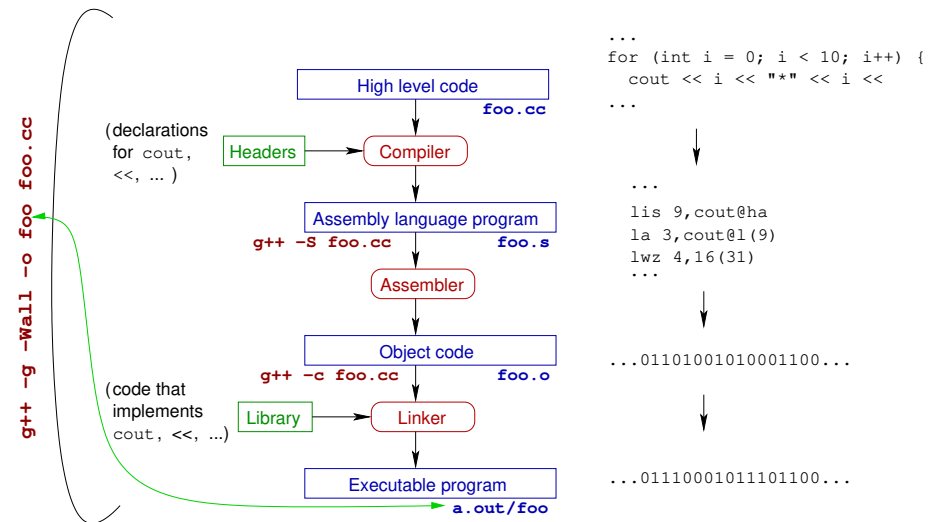
```
#include "tcp-util.h"
```

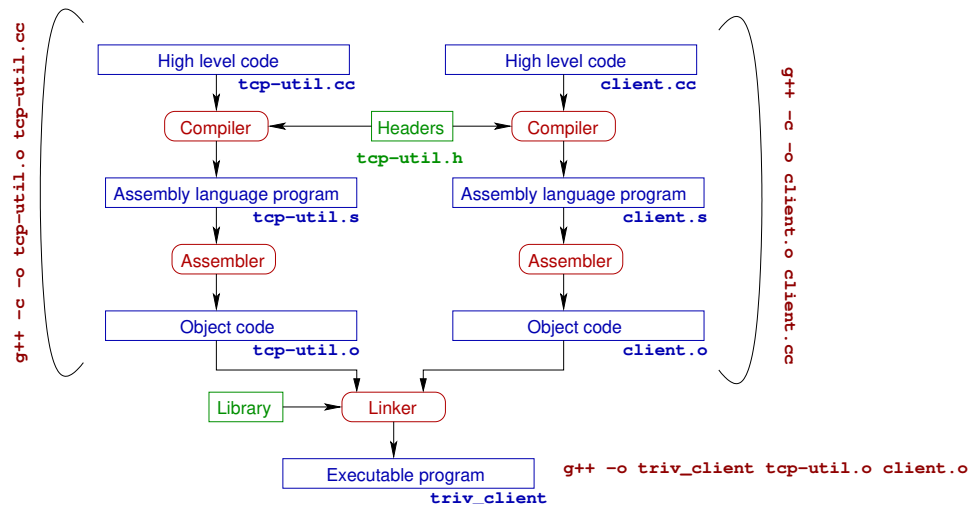
  - Then `tcp-util.cc` and `main.cc` will be compiled and linked together
  - We can automate this process by encoding the recipe into a **makefile**

# BRINGING YOUR PROGRAM TO LIFE



# BRINGING YOUR PROGRAM TO LIFE (CONT'D)





- A makefile contains recipes for compiling multiple file programs
- A makefile contains **macrodefinitions**, e.g.,  

```
# this is a comment
CXX = g++
CXXFLAGS = -g -Wall
```
- Then we have **rules** of the form:

```
target : [source1 ] [source2 ] [source3 ]
        command1
        command2
        command3
        ...
```

Exactly one TAB on each line here!

- A **target** is the name of the file to be produced
  - It is produced by executing the corresponding **commands**
- The **sources** are the files needed to produce the target (if any)
  - They form a **dependency tree**



- Sample makefile:

```
all: triv_client

tcp-utils.o: tcp-utils.h tcp-utils.cc
    $(CXX) $(CXXFLAGS) -c -o tcp-utils.o tcp-utils.cc

client.o: tcp-utils.h client.cc
    $(CXX) $(CXXFLAGS) -c -o client.o client.cc

triv_client: client.o tcp-utils.o
    $(CXX) $(CXXFLAGS) -o triv_client client.o tcp-utils.o

clean:
    rm -f triv_client *~ *.o *.bak core \#*
```

- Suppose you type **make target** in some directory **d**
  - **make** without arguments produces the **first** target in the makefile
- The command looks for a file called **Makefile** in **d** and follows all the necessary rules therein along the dependency tree to produce the file **target**
- All the targets needed by **target** (based on said dependency tree) are also made, **unless they are up to date** (decision based on modification times)