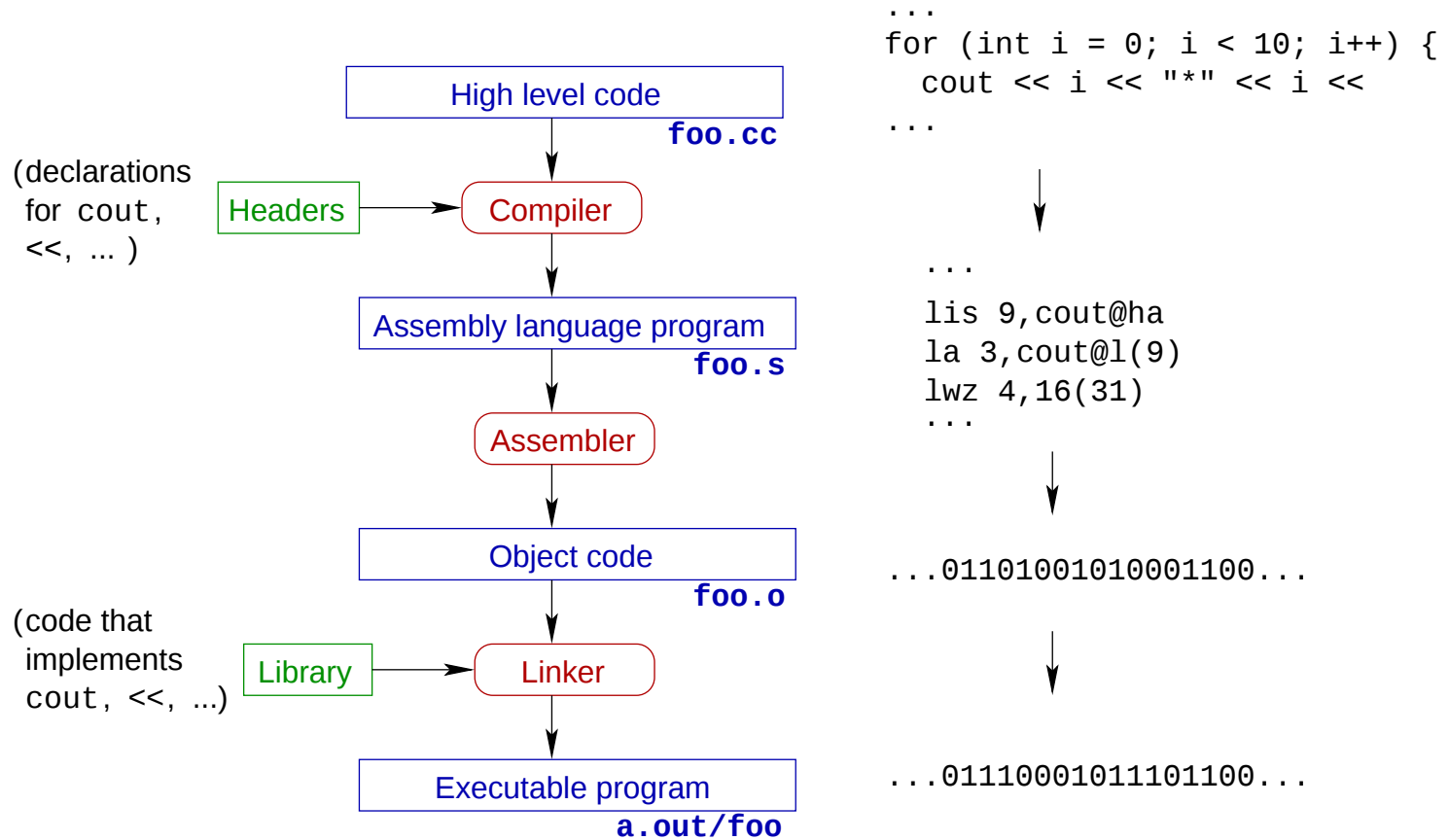


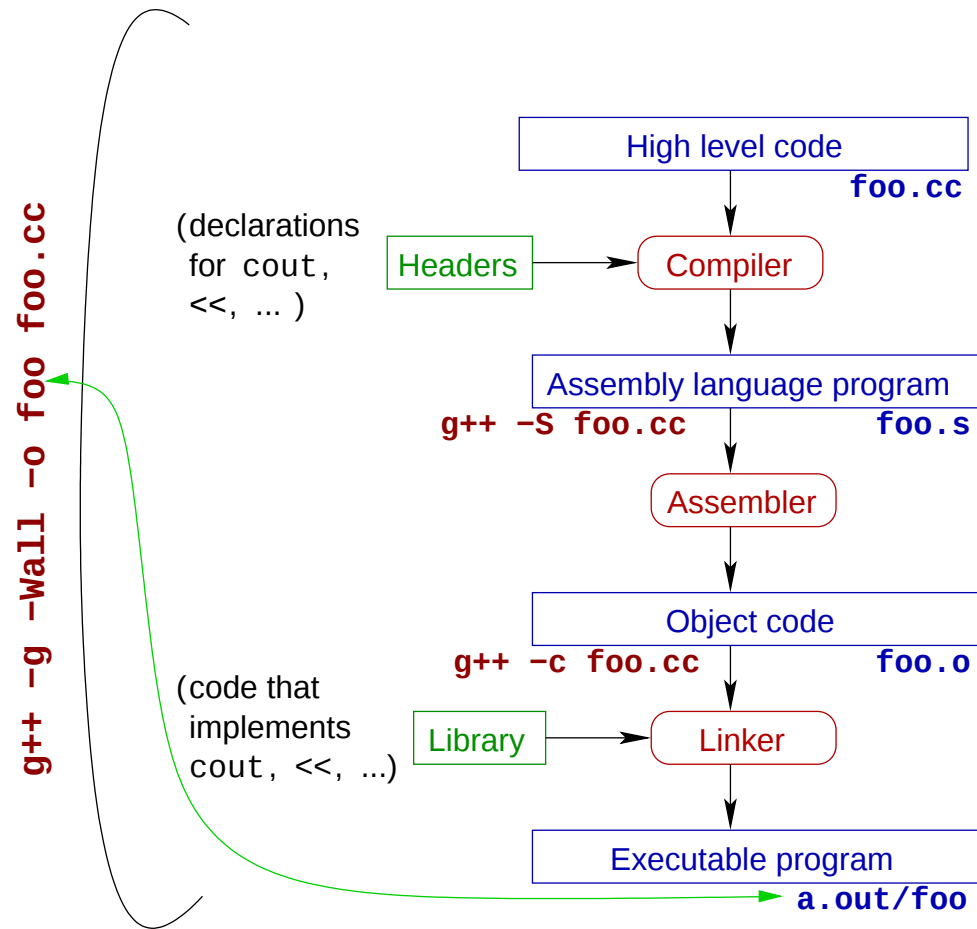
SPLITTING THE CODE

- Sometimes we like to split our program into multiple files (or **modules**).
- Advantages: encapsulation, reusability, size.
 - We can also reduce compilation time.
- A module consists in 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 do `#include "tcp-util.h"`
 - Then `tcp-util.cc` and `main.cc` will be compiled and linked together.
 - * We use for this purpose a **makefile**.

BRINGING YOUR PROGRAM TO LIFE



BRINGING YOUR PROGRAM TO LIFE



```
...  
for (int i = 0; i < 10; i++) {  
    cout << i << "*" << i <<  
...  
...
```



```
...  
lis 9,cout@ha  
la 3,cout@l(9)  
lwz 4,16(31)  
...
```

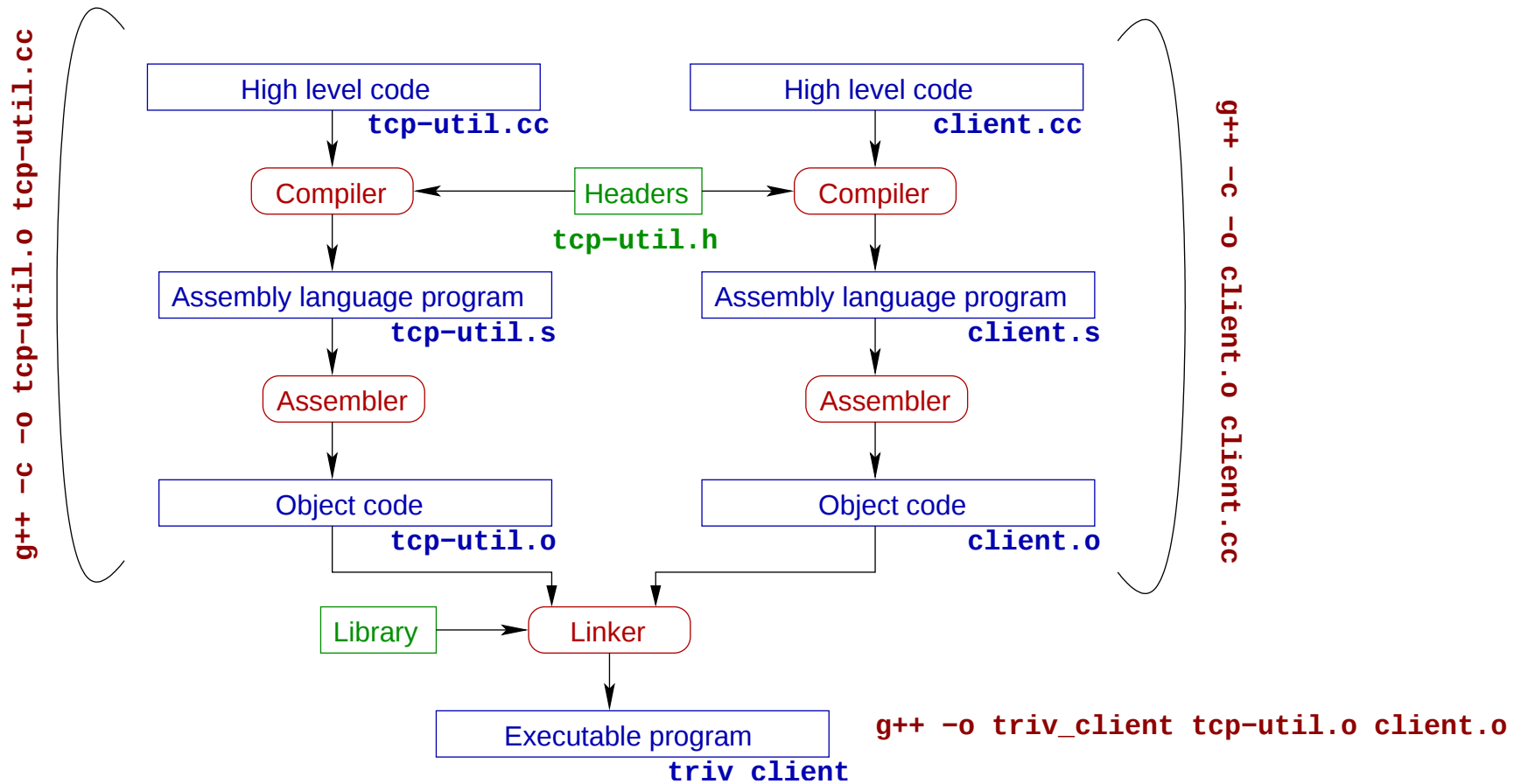


```
...01101001010001100...
```



```
...01110001011101100...
```

PUTTING MANY MODULES TOGETHER



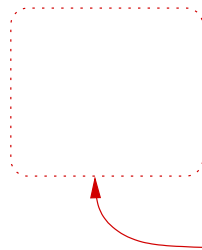
MAKEFILES

- 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)

MAKEFILES (CONT'D)

- Sample of rules:

```
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 \#*
```

- 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 produces the file *target*.
- All the targets needed by *target* are also made, **unless they are up to date**.