# Makefile Utility

Công cụ hỗ trợ biên dịch project

# Những vấn đề khi biên dịch

- Một chương trình đơn giản => chỉ có một vài file
- Một chương trình “không đơn giản”
  - Nhiều dòng lệnh
  - Nhiều module
  - Nhiều người tham gia viết

# Những vấn đề khi biên dịch

- Vấn đề xảy ra:
  - Khó quản lý một file lớn (cả người và máy)
  - Mỗi thay đổi cần thời gian biên dịch lâu
  - Nhiều người lập trình không thể thay đổi cùng một file đồng thời
  - Chương trình được phân ra thành nhiều module

# Những vấn đề khi biên dịch

- Giải pháp: chia project ra thành nhiều file
- Mục tiêu:
  - Chia thành các module một cách đúng đắn
  - Thời gian biên dịch ngắn nếu có sự thay đổi
  - Dễ dàng bảo trì cấu trúc project và sự phụ thuộc

# Makefile là gì?

- Công cụ hỗ trợ biên dịch
- Chỉ biên dịch những phần cần thiết
- Biên dịch trên nhiều platform khác nhau

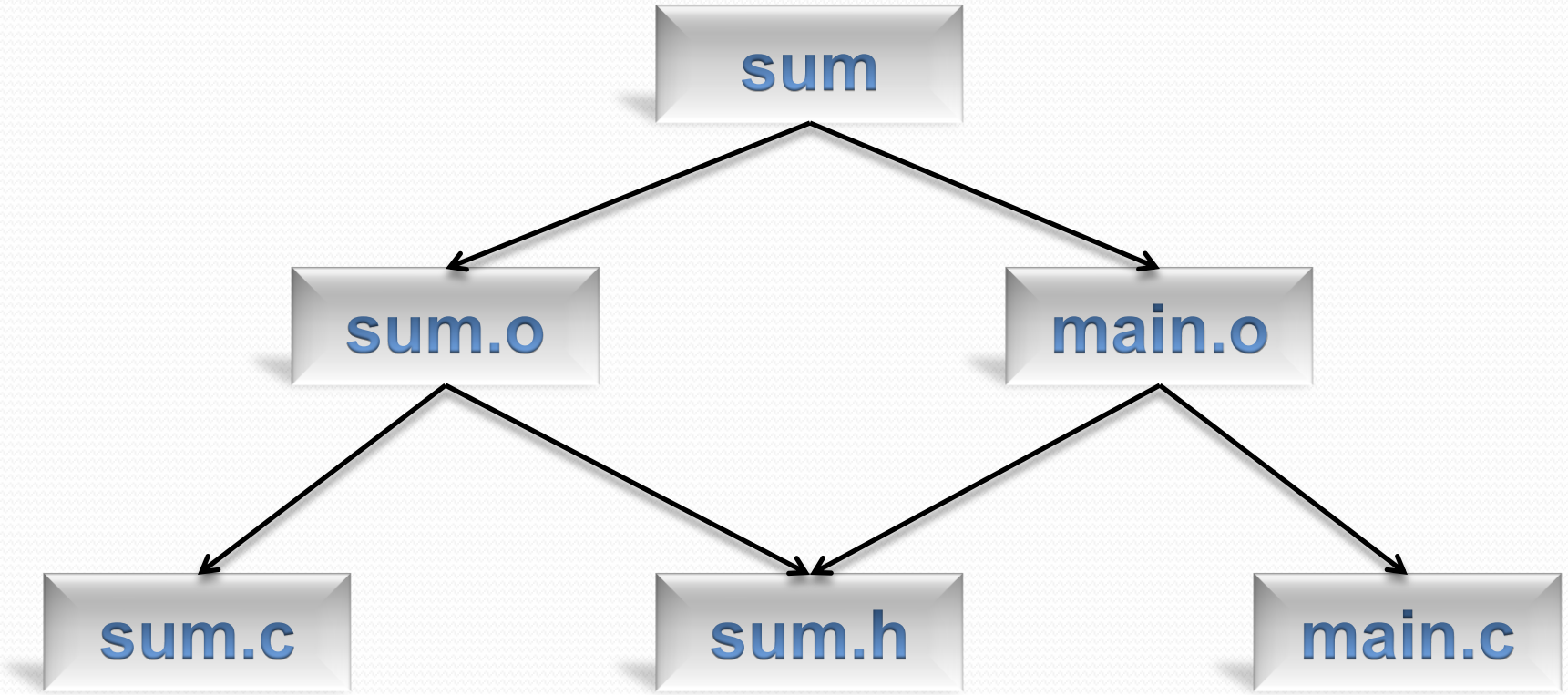
# Bảo trì project

- Trên Unix được thực hiện bởi Makefile
- Makefile là một file dạng script chứa các thông tin:
  - Cấu trúc project (file, sự phụ thuộc)
  - Các lệnh để tạo file
- Lệnh `make` sẽ đọc nội dung Makefile, hiểu kiến trúc của project và thực thi các lệnh
- Makefile không giới hạn trong ngôn ngữ C/C++

# Cấu trúc project

- Cấu trúc và sự phụ thuộc của project có thể được biểu diễn bằng một DAG (Directed Acyclic Graph)
- Thí dụ:
  - Chương trình chứa 3 file: main.c, sum.c, sum.h
  - File sum.h được dùng bởi cả 2 file main.c và sum.c
  - File thực thi là sum

# Cấu trúc project



# Nội dung Makefile

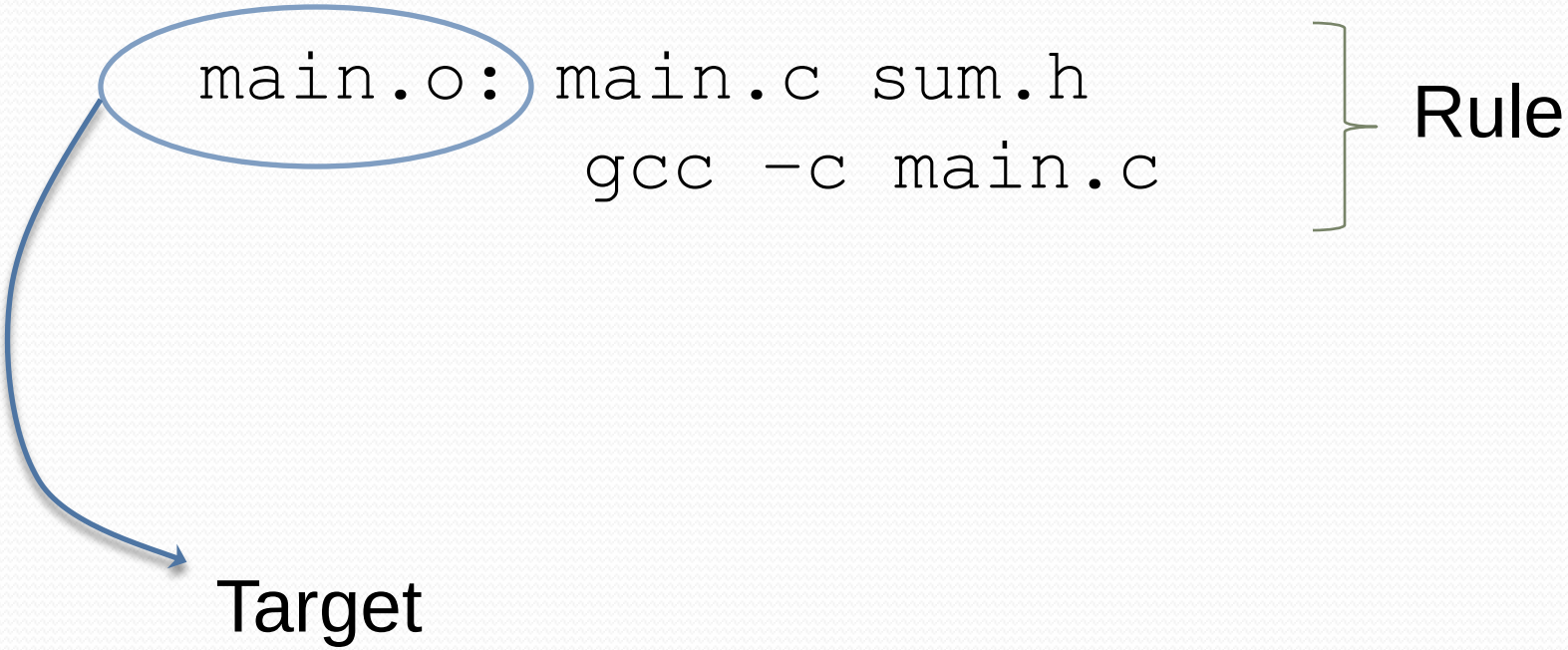
```
sum: main.o sum.o
	gcc -o sum main.o sum.o

main.o: main.c sum.h
	gcc -c main.c

sum.o: sum.c sum.h
	gcc -c sum.c
```

# Cú pháp

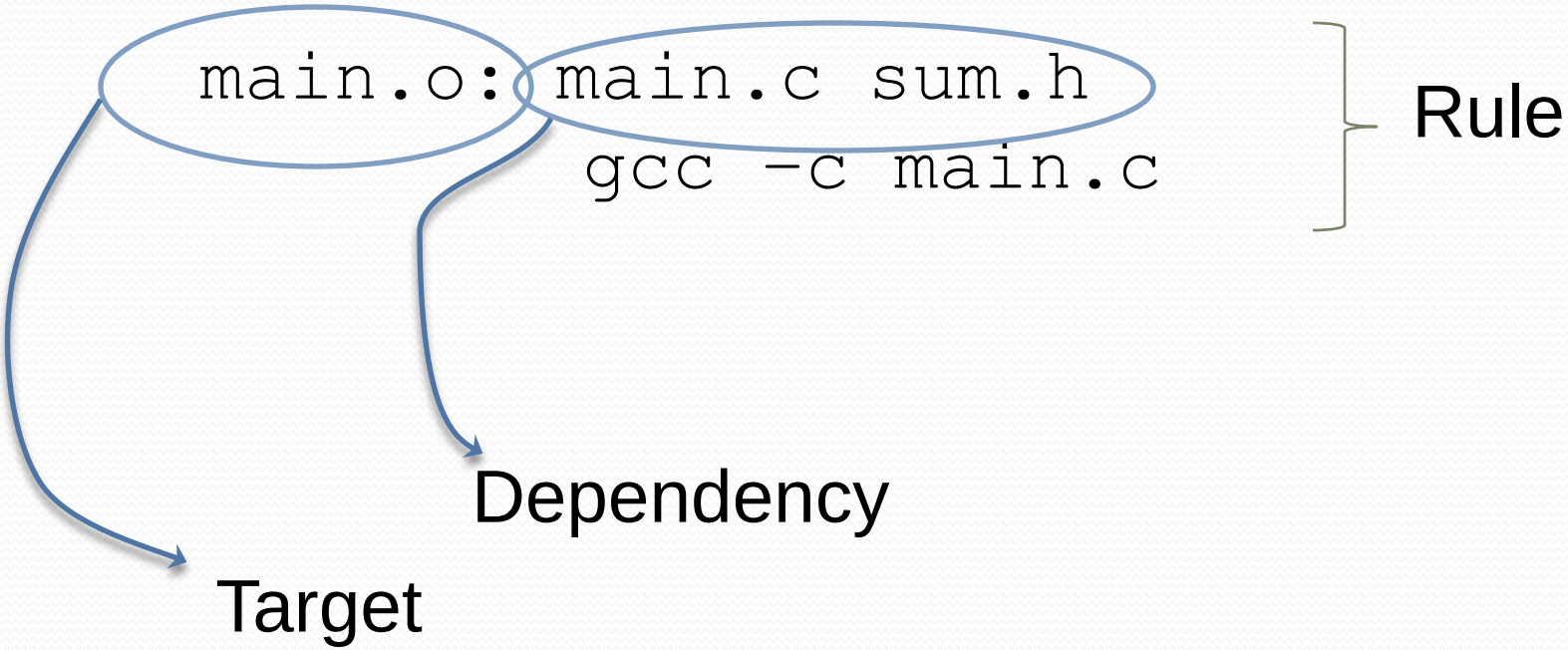
```
main.o: main.c sum.h
        gcc -c main.c
```

Rule

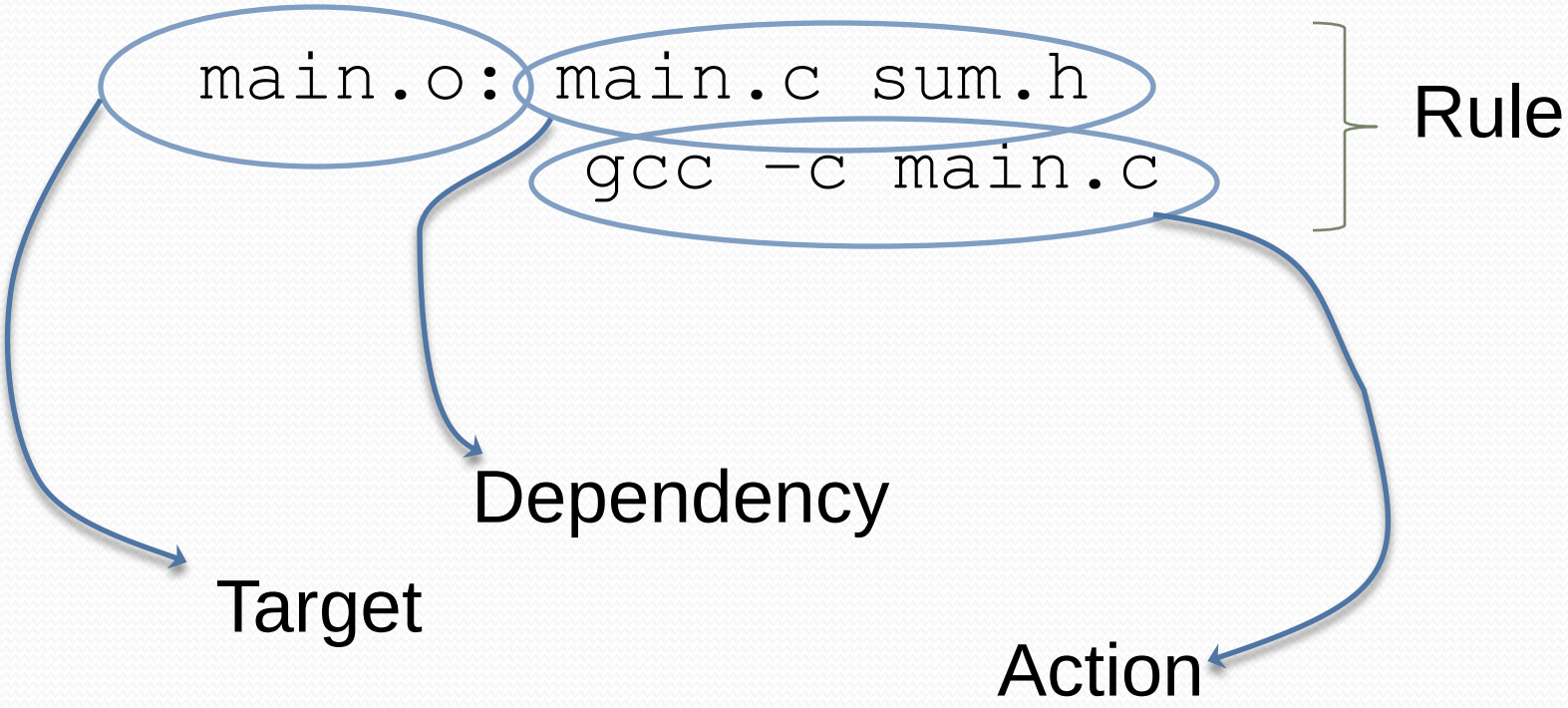
# Cú pháp



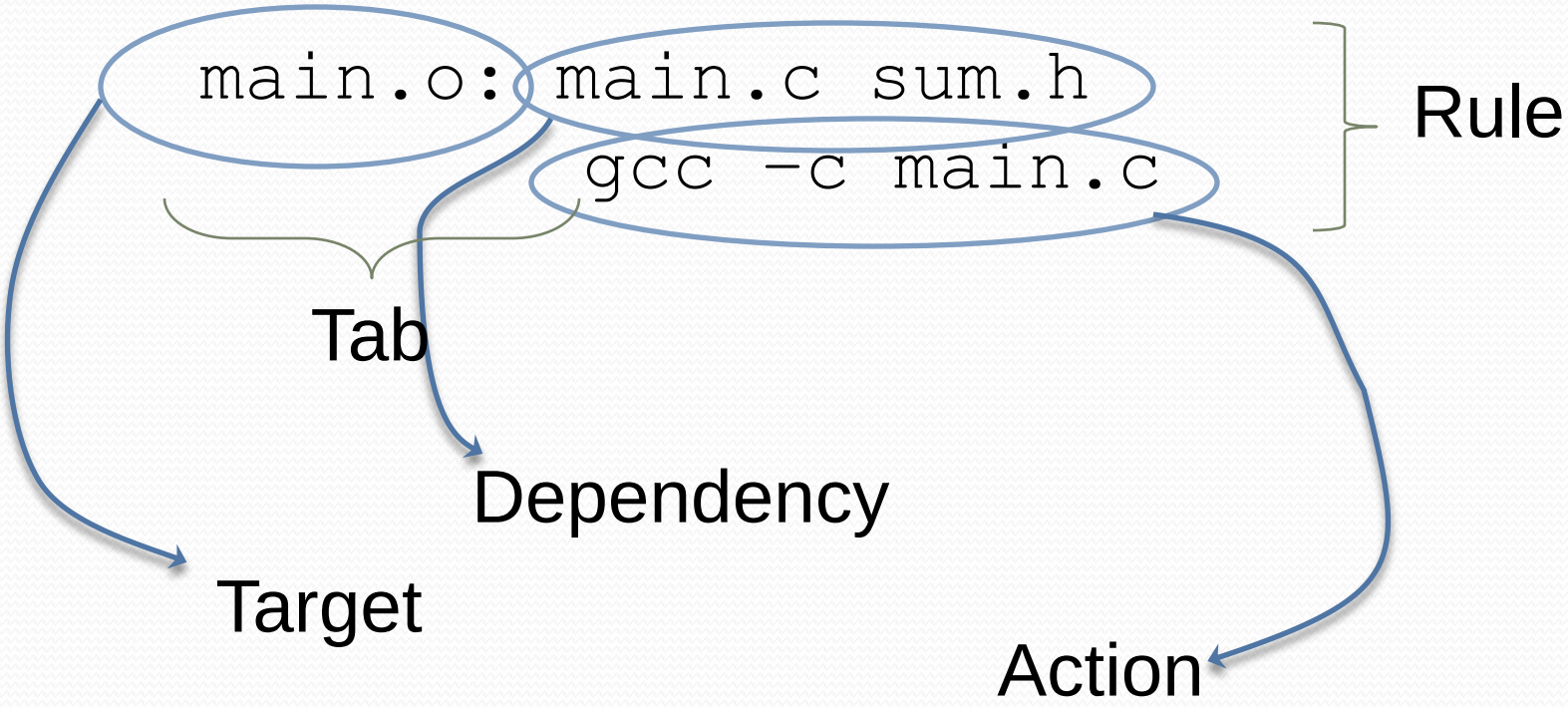
# Cú pháp



# Cú pháp



# Cú pháp


main.o: main.c sum.h
gcc -c main.c
Rule
Tab
Dependency
Target
Action

# Định nghĩa Target

- Thí dụ về một dòng target với các phụ thuộc:
  - sum.o: sum.c sum.h
  - all: sum
- Và thí dụ về một target không có phụ thuộc:
  - clean:
- Các target không bắt đầu bằng khoảng trắng hoặc TAB

# Định nghĩa Action

- Mỗi target sẽ kèm theo 0 hoặc nhiều action.
- Mỗi dòng chứa action phải bắt đầu bằng TAB
- Thí dụ:
  - <TAB> gcc –o sum sum.o main.o
  - <TAB> rm –fr ${OBJ}
  - <TAB> gcc –c sum.c

# Phép gán

- Makefile cho phép định nghĩa các biến và gán giá trị cho nó
  - CC = gcc
  - LIBS = “-lncurses –lm”
  - echo ${CC}

# Phép gán

- Thí dụ:

```
CC = gcc
LIBS = "-lpthread"
TARGET = prog
OBJS = main.o thread.o

${TARGET}: ${OBJS}
    ${CC} ${LIBS} -o ${TARGET} ${OBJS}

.c.o:
    ${CC} -c $<
```

# Makefile tương đương

- Có thể viết Makefile với sự phụ thuộc ngắn gọn hơn dùng những macro có sẵn hỗ trợ:

```
sum: main.o sum.o
	gcc -o $@ main.o sum.o

main.o sum.o: sum.h
	gcc -c $*.c
```

# Hoạt động của Makefile

- Xây dựng cây phụ thuộc của project
- Target của rule đầu tiên phải được tạo
- Dò theo cây phụ thuộc để tìm xem target có cần phải tạo lại không. Nếu target *cũ* hơn các file phụ thuộc thì cần phải tạo lại.
- Nếu có một target được tạo lại thì các target của nó cũng được tạo lại theo.

# Hoạt động của Makefile

- Hoạt động của make đảm bảo sự biên dịch ít nhất nếu như cấu trúc của project được mô tả đúng.
- KHÔNG NÊN viết như sau:

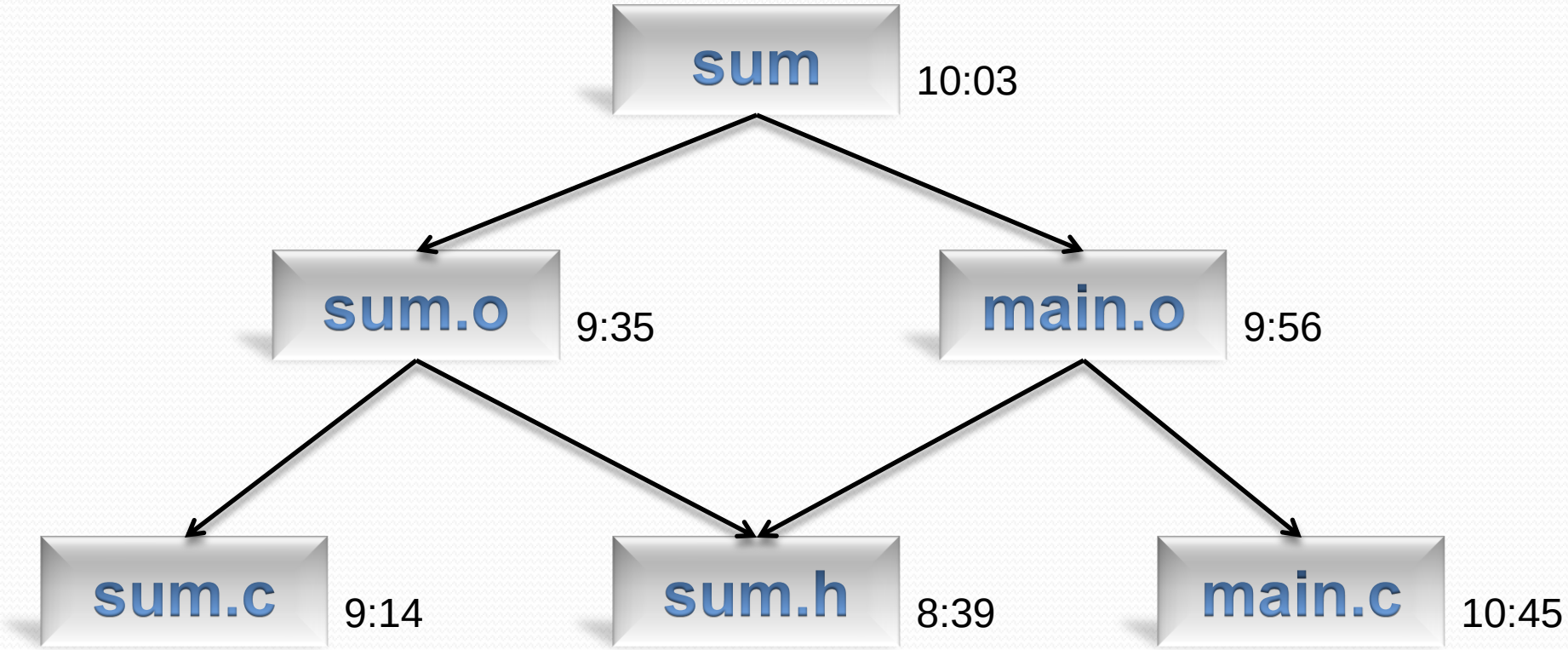
```
prog: main.c sum1.c sum2.c
	gcc –o prog main.c sum1.c sum2.c
```

Vì khi đó toàn bộ project sẽ được dịch lại nếu có sự thay đổi trong project.

# Thí dụ

| File | Last Modified |
| --- | --- |
| sum | 10:03 |
| main.o | 09:56 |
| sum.o | 09:35 |
| main.c | 10:45 |
| sum.c | 09:14 |
| sum.h | 08:39 |

# Thí dụ



# Thí dụ

- Các tác vụ được thực hiện:

```
gcc –c main.c
gcc –o sum main.o sum.o
```

- main.o phải được biên dịch lại (vì main.c mới hơn)
- Do đó, main.o sẽ mới hơn sum, và sum sẽ được tạo lại