QUESTIONS

1. What are the advantages of Makefile? Give examples?

- Việc sử dụng Makefile giúp quản lý, biên dịch các chương trình, dự án với nhiều file, có độ phức tạp cao. Việc thêm các "Rule" vào trong Makefile như các chỉ dẫn rằng chương trình sẽ được biên dịch ra sao, giúp ta tiết kiệm thời gian và công sức, hạn chế sai sót, đặc biệt là khi có sự thay đổi trong các file, source code,… hoặc các yêu cầu bảo trì sau này. Makefile sẽ tự động biên dịch lại những tệp bị thay đổi.
- Giúp cho việc trình bày code, quá trình biên dịch trở nên rõ ràng và có hệ thống hơn, thuận tiện cho việc đọc và sửa lỗi sau này.
- Ví dụ rằng, chúng ta có một tệp rất nhiều file có sự ràng buộc với nhau. Việc thay đổi dữ liệu trong một số file có thể dẫn đến việc biên dịch lại cả một số thành phần, hoặc cả chương trình. Tuy nhiên điều này không dễ để thực hiện vì chúng ta rất khó có thể nắm rõ được cần phải biên dịch lại những gì… Và hơn nữa, sẽ là không tối ưu nếu sau mỗi lần sửa đổi chúng ta đều phải biên dịch một cách thủ công lại gần như cả chương trình. Sử dụng Makefile sẽ giúp tiết kiệm tối đa thời gian cũng như công sức cho vấn đề này, và đặc biệt giảm thiểu đến mức tối thiểu những sai sót có thể xảy ra so với việc biên dịch lại từng file như đề cập.

2. In case of source code files located in different places, how can we write a Makefile?

- Lấy Problem_3 làm ví dụ, giả sử ta có các source code được lưu trong các folder theo sơ đồ sau:

-- test
    -- test_5
        -- Header
            -- factorial.h
            -- readline.h
    -- test_6
        -- main.c
        -- Source
            -- factorial.c
            -- readline.c

- Lúc này, source code nằm ở các directory khác nhau. Ta sẽ tạo Makefile tại "test_6" như sau:

```
CC = gcc
CFLAGS = -c -I../test_5/Header
OBJ = main.o readline.o factorial.o
H = ../test_5/Header
S = Source

myfactorial: $(OBJ)
        $(CC) $^ -o $@
main.o: main.c $(H)/readline.h $(H)/factorial.h
        $(CC) $(CFLAGS) $< -o $@
readline.o: $(S)/readline.c $(H)/readline.h
        $(CC) $(CFLAGS) $< -o $@
factorial.o: $(S)/factorial.c $(H)/factorial.h
        $(CC) $(CFLAGS) $< -o $@
clean:
        rm -f *.o myfactorial
```

- Ta sẽ chỉ rõ directory của từng Dependency, đồng thời sử dụng thêm flag -I trên các command line để chỉ ra rằng cần tìm kiếm các file .h ở Header trong folder test_5 (-I../test_5/Header).

- Kết quả sau khi biên dịch:

```
quockhanh@Khanh-VirtualBox:~/test/test_6$ cat Makefile
CC = gcc
CFLAGS = -c -I../test_5/Header
OBJ = main.o readline.o factorial.o
H = ../test_5/Header
S = Source

myfactorial: $(OBJ)
        $(CC) $^ -o $@
main.o: main.c $(H)/readline.h $(H)/factorial.h
        $(CC) $(CFLAGS) $< -o $@
readline.o: $(S)/readline.c $(H)/readline.h
        $(CC) $(CFLAGS) $< -o $@
factorial.o: $(S)/factorial.c $(H)/factorial.h
        $(CC) $(CFLAGS) $< -o $@
clean:
        rm -f *.o myfactorial
quockhanh@Khanh-VirtualBox:~/test/test_6$ make
gcc -c -I../test_5/Header main.c -o main.o
gcc -c -I../test_5/Header Source/readline.c -o readline.o
gcc -c -I../test_5/Header Source/factorial.c -o factorial.o
gcc main.o readline.o factorial.o -o myfactorial
quockhanh@Khanh-VirtualBox:~/test/test_6$ ./myfactorial
Bach Khoa
-1
quockhanh@Khanh-VirtualBox:~/test/test_6$ ./myfactorial
6
720
quockhanh@Khanh-VirtualBox:~/test/test_6$
```

3. What the output will be at LINE A? Explain your answer.

- Chương trình sẽ in ra:

PARENT: value = 5

- Do process cha và process con làm việc trên hai không gian địa chỉ riêng biệt nên việc thay đổi giá trị của value ở process con không làm ảnh hưởng đến value ở process cha. Cho nên với pid > 0, dẫn đến việc process cha vẫn sẽ cho in ra giá trị value = 5.

- Thử nghiệm:

```
quockhanh@Khanh-VirtualBox:~/test/test_4$ make
make: 'test_4' is up to date.
quockhanh@Khanh-VirtualBox:~/test/test_4$ ./test_4
PARENT: value = 5quockhanh@Khanh-VirtualBox:~/test/test_4$
```