

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA



HỆ ĐIỀU HÀNH (CO2018)

LAB 1

C Programming on Linux, Mac OS X Process

GVHD: Nguyễn Quang Hùng

Sinh viên: Hoàng Tiến Hải - 2011152

Tp. Hồ Chí Minh, Tháng 1/2022



Mục lục

1	INTRODUCTION	2
2	PRACTICE	2
3	EXERCISES	2
3.1	Questions	2
3.2	Basic commands	3
3.3	Programming exercises	3
	Tài liệu	8

1 INTRODUCTION

2 PRACTICE

3 EXERCISES

3.1 Questions

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

– Các lợi ích của việc sử dụng Makefile:

- Giúp người dùng compile chương trình dễ dàng với các rule được viết ra trong file quy định.
- Kiểm soát được thứ tự, biên dịch file nào cần thiết.
- Giúp nhanh hơn trong việc kiểm soát các file biên dịch, chỉ biên dịch những file có thay đổi giúp giảm thời gian biên dịch.
- Thực hiện mọi thứ một cách tự động bằng cách sử dụng từ khoá make.
- etc ...

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

– Ta có thể tạo Makefile đối với trường hợp source code đã được lưu trữ ở một đường dẫn khác bằng cách khai báo các tham số có giá trị là đường dẫn đến các file cần biên dịch.

– Ví dụ: Tập tin main.c là source code được lưu trữ trong folder source, tập tin hello.c và hello.h là hai file được lưu trữ trong folder header. Các tạo Makefile như sau:

- Tạo một tập tin Makefile bằng lệnh **vim**

```
~/ $ vim Makefile
```

- Định nghĩa đường dẫn tới tập tin source code và tập tin header.

```
PWD := .  
SRC := $(PWD)/source  
HEAD := $(PWD)/header
```

- Hoàn thiện biên dịch trong Makefile.

```
PWD := .  
SRC := $(PWD)/source  
HEAD := $(PWD)/header  
  
all: main.o hello.o  
    gcc main.o hello.o -o main  
  
main.o: $(SRC)/main.c $(HEAD)/hello.h  
    gcc -c $(SRC)/main.c  
  
hello.o: $(HEAD)/hello.c $(HEAD)/hello.h  
    gcc -c $(HEAD)/hello.c
```

– Sau khi gọi lệnh **make all** trong terminal. Ta sẽ có tập tin thực thi **main**.

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

```
#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>
#include <unistd.h>

int value = 5;
int main()
{
    pid_t pid;
    pid = fork();
    if (pid == 0) { /* child process */
        value += 15;
        return 0;
    }
    else if (pid > 0) { /* parent process */
        wait(NULL);
        printf("PARENT: _value_=%d", value); /* LINE A */
        return 0;
    }
}
```

– Kết quả ở dòng LINE A là "PARENT: value = 5".

– Khi chương trình được biên dịch và thực thi, biến value toàn cục sẽ được khởi tạo với giá trị là 5. Trong lúc thực thi chương trình, câu lệnh **fork()** sẽ tạo ra một child process. Child process này sẽ được cấp phát một PCB (Process control block) với các vùng nhớ, dữ liệu mới và sao chép giá trị từ parent process và pid bằng 0. Vì vậy khi thay đổi giá trị value ở child process sẽ không làm thay đổi giá trị value ở parent process.

3.2 Basic commands

3.3 Programming exercises

1. **PROBLEM 1. Write factorial.c to implement function factorial(): the function get an integer and return its factorial.**

– Trong tệp tin factorial.c, ta chỉ **include** tệp tin factirial.h để hiện thực hàm factorial được định nghĩa trong tệp tin header. Sử dụng đệ quy để tính factorial của số **aNumber**.

```
#include "factorial.h"

int factorial(const int aNumber)
{
    if (aNumber == 0) return 1;
    int tmp = aNumber - 1;
    return aNumber * factorial(tmp); //De quy
}
```

– Kiểm thử chương trình:

2. **PROBLEM 2. Write readline.c to implement read_line(): read_line() gets data from stdin (keyboard), line-by-line. The content from stdin will be recorded on the parameter of this function named str. The result of read_line() indicates that whether the line is an integer or not.**

– Ta hiện thực một hàm **strCopy** để copy từ string src sang string des. Điều này để chuyển dữ liệu nhập từ bàn phím vào chuỗi truyền vào.

```
~/probl/ $ cat Makefile
all: problem1.o factorial.o
    gcc problem1.o factorial.o -o prob1

problem1.o: problem1.c factorial.h
    gcc -c problem1.c

factorial.o: factorial.c factorial.h
    gcc -c factorial.c

clean:
    rm -f *.o prob1~/probl/ $
~/probl/ $ make all
gcc -c problem1.c
gcc -c factorial.c
gcc problem1.o factorial.o -o prob1
~/probl/ $ ./prob1
Enter a number: 7
My factorial is: 5040
```

Hình 1

```
void strCopy(char *src, char *des)
{
    int i = 0;
    while (src[i] != '\0')
    {
        des[i] = src[i];
        i++;
    }
}
```

– Hàm `read_line()` sẽ yêu cầu người dùng nhập vào một chuỗi. Sau đó duyệt một vòng lặp while từ đầu đến cuối chuỗi, kiểm tra nếu có ký tự nào khác số thì sẽ trả về giá trị 0. Ngược lại khi hết chuỗi sẽ trả về giá trị 1.

```
#include <stdio.h>

void strCopy(char *src, char *des)
{
    int i = 0;
    while (src[i] != '\0')
    {
        des[i] = src[i];
        i++;
    }
}

int read_line(char *str)
{
    char line[512];
    fgets(line, 512, stdin);
    strCopy(line, str);
    int i = 0;
    while (str[i] != '\n' && str[i] != '\0')
    {
        if (str[i] < '0' || str[i] > '9' || str[i] == '\0') return 0;
    }
}
```

```

        ++i;
    }
    return 1;
}

```

– Kiểm thử chương trình:

```

~/prob2/ $ cat Makefile
all: problem2.o readline.o
    gcc problem2.o readline.o -o prob2

problem1.o: problem2.c readline.h
    gcc -c problem2.c

readline.o: readline.c readline.h
    gcc -c readline.c

clean:
    rm -f *.o prob2~/prob2/ $
~/prob2/ $ make all
clang -ggdb3 -O0 -std=c11 -Wall -Werror -Wextra -Wno-sign-compare -Wno-unused-parameter -Wno-unused-variable -Wshadow -c -
o problem2.o problem2.c
gcc -c readline.c
gcc problem2.o readline.o -o prob2
~/prob2/ $ ./prob2
Hello World
0
245
1

```

Hình 2

3. **PROBLEM 3.** Write `main.c` to create an executable file named `myfactorial` that reads input from `stdin` line by line and compute factorial if the line is an integer (each line does not exceed 50 letters). Then print factorial if the line is an integer else print -1.

– Ta sẽ lần lượt include `factorial.h` và `readline.h` đã được hiện thực từ trước để sử dụng.
– Thêm vào đó, ta viết thêm các hàm hỗ trợ cho việc hiện thực:

- `int power(int a, int b)`: Trả về giá trị của a lũy thừa b .
- `int sizeStr(char *str)`: Trả về kích thước của một chuỗi.
- `int strToInt(char *str)`: Chuyển một chuỗi string sang thành một số nguyên.

– Ở hàm `main`, ta sẽ yêu cầu người dùng nhập một chuỗi vào và lưu trong biến `tmp`. Sau đó dùng hàm `read_line()` để kiểm tra chuỗi vừa nhập có là chuỗi hay không. Nếu có tiến hành tính giai thừa thông qua hàm `factorial`, ngược lại in ra -1.

```

#include <stdio.h>
#include "readline.h"
#include "factorial.h"

int power(int a, int b)
{
    if (b == 0) return 1;
    if (a == 0 || b == 1) return a;
    return a * power(a, b-1);
}

int sizeStr(char *str)
{
    int count = 0;
    while (str[count] != '\n' && str[count] != '\0')
    {

```

```
        ++count;
    }
    return count;
}

int strToInt(char *str)
{
    int res = 0;
    int size = sizeStr(str);
    int tmp = size;
    for (int i = 0; i < size; i++)
    {
        res += (str[i] - '0') * power(10, --tmp);
    }
    return res;
}

int main (int argc , char *argv []){
    while (1)
    {
        char tmp[1000];
        if (read_line(tmp) == 0)
        {
            printf("-1\n");
        }
        else
        {
            printf("My_factorial_is:_%d\n", factorial(strToInt(tmp)));
        }
    }
    return 0;
}
```

– Tạo một makefile để chạy chương trình:

```
all: problem3.o factorial.o readline.o
    gcc problem3.o factorial.o readline.o -o myfactorial

problem3.o: problem3.c factorial.h readline.h
    gcc -c problem3.c

factorial.o: factorial.c factorial.h
    gcc -c factorial.c

readline.o: readline.c readline.h
    gcc -c readline.c

clean:
    rm -f *.o myfactorial
```

– Kiểm thử chương trình:

4. **Problem 4** Given a file named "numbers.txt" containing multiple lines of text. Eachline is a non-negative integer. Write a C program that reads integers listed in this file and

```
~/prob3/ $ vim Makefile
~/prob3/ $ make all
gcc -c problem3.c
gcc -c factorial.c
gcc -c readline.c
gcc problem3.o factorial.o readline.o -o myfactorial
~/prob3/ $ ./myfactorial
Hello World
-1
5
My factorial is: 120
```

Hình 3

stores them in an array (or linked list). The program then uses the `fork()` system call to create a child process. The parent process will count the numbers of integers in the array that are divisible by 2. The child process will count numbers divisible by 3. Both processes then send their results to the stdout.

– Trước tiên ta sẽ tiến hành đọc tệp tin `numbers.txt` bằng lệnh **fopen**, sau đó đọc từng dòng một bằng lệnh **fgets** và lưu số vừa đọc được vào một mảng `arr` đã tạo từ trước, đồng thời tìm size của mảng.

```
in = fopen("numbers.txt", "r");
int size = 0;
while (fgets(line, sizeof(line), in) != NULL)
{
    int val = atoi(line);
    arr[size++] = val;
}
```

– Sau đó gọi lệnh **pid_t pid = fork()** sẽ tạo ra một child process có pid được lưu trong biến `pid` bằng 0, còn parent process sẽ có `pid > 0`. Vì vậy kiểm tra giá trị của `pid` ta sẽ điều khiển để duyệt mảng và đếm giá trị chia hết cho 3 đối với child process và các giá trị chia hết cho 2 đối với parent process. Nếu `pid < 0`, hàm sẽ báo lỗi do không thể tạo thêm process.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main()
{
    FILE *in;
    char line[512];
    int arr[1000];
    in = fopen("numbers.txt", "r");
    int size = 0;
    while (fgets(line, sizeof(line), in) != NULL)
    {
        int val = atoi(line);
        arr[size++] = val;
    }

    int count = 0;

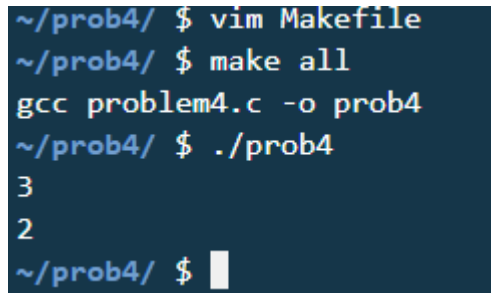
    pid_t pid = fork();

    if (pid == 0)
```



```
{  
    int i = 0;  
    for (int i = 0; i < size; i++)  
    {  
        if (arr[i] % 3 == 0) count++;  
    }  
  
    else if (pid > 0)  
    {  
        int i = 0;  
        for (int i = 0; i < size; i++)  
        {  
            if (arr[i] % 2 == 0) count++;  
        }  
    }  
    else printf("Error_to_create_child_process\n");  
  
    printf("%d\n", count);  
  
    return 0;  
}
```

– Kiểm thử chương trình:



```
~/prob4/ $ vim Makefile  
~/prob4/ $ make all  
gcc problem4.c -o prob4  
~/prob4/ $ ./prob4  
3  
2  
~/prob4/ $
```

Hình 4

Tài liệu

- [Dal] Dalgaard, P. *Introductory Statistics with R*. Springer 2008.
- [K-Z] Kenett, R. S. and Zacks, S. *Modern Industrial Statistics: with applications in R, MINITAB and JMP*, 2nd ed., John Wiley and Sons, 2014.
- [Ker] Kerns, G. J. *Introduction to Probability and Statistics Using R*, 2nd ed., CRC 2015.