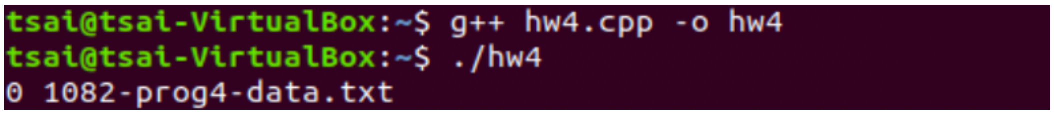
OS — hw4 學號:1071552 姓名:蔡曜亦

1. 程式的設計理念、程式如何編譯，以及如何操作:

RMS：依據頻率高低決定優先權，週期短(頻率高)有高優先權，週期長(頻率低)較低優先權。

EDF：依據誰的 deadline 先到，誰的優先權就越高。

編譯及操作: (1) g++ 檔名.cpp –o 檔名 (2) ./檔名 (3)分為選項 0、選項 1、選項 2 再打 txt 檔名


```
tsai@tsai-VirtualBox:~$ g++ hw4.cpp -o hw4
tsai@tsai-VirtualBox:~$ ./hw4
0 1082-prog4-data.txt
```

2. 完成部分:

a. 基本功能

i. 選項 0: 能正確以 RMS 排程 2 ≤ n ≤ 5 個 processes，輸出模擬結果。此項目最多得 40 分。

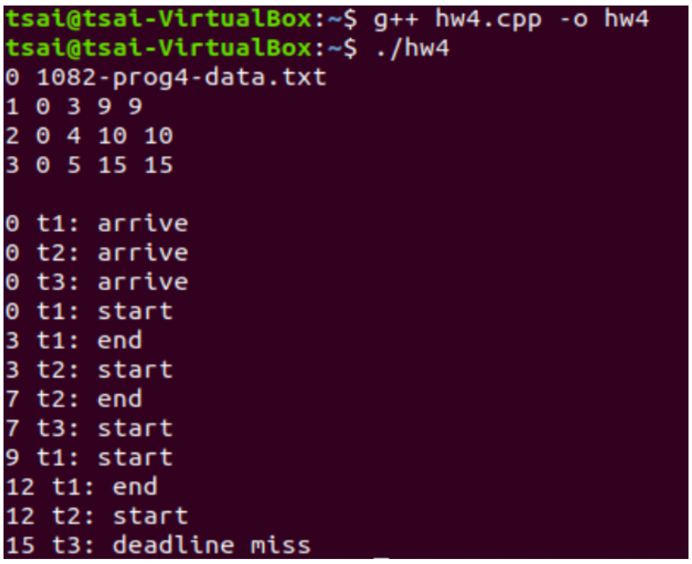
ii. 選項 1: 能正確以 EDF 排程 2 ≤ n ≤ 5 個 processes，輸出模擬結果。

b. 進階功能

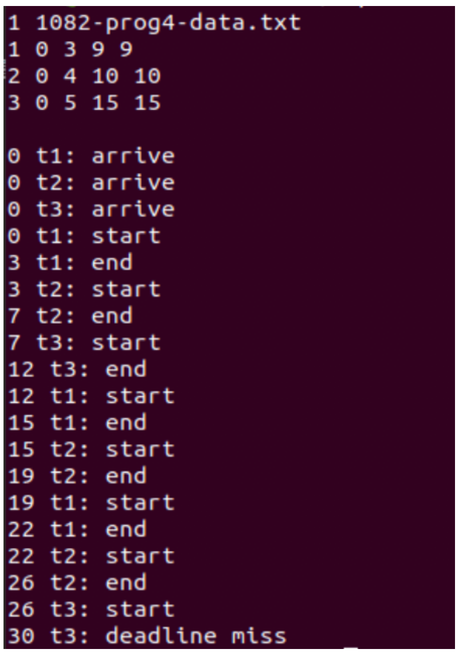
i. 選項 2: 模擬 EDF，但 D 與 T 可能不同，D ≤ T。模擬 2 ≤ n ≤ 5 個 processes 排程結果。

結果:

基本功能(選項 0):

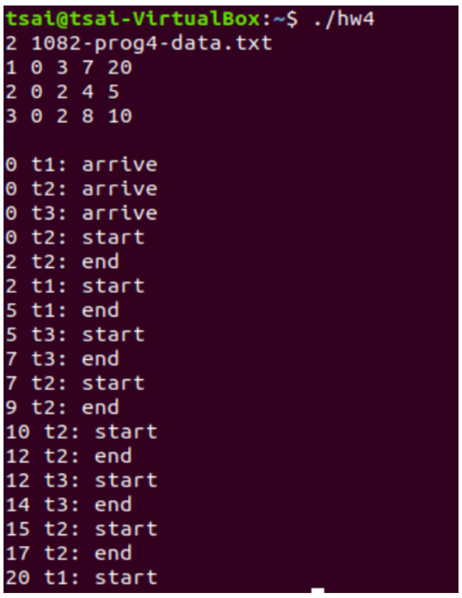

```
tsai@tsai-VirtualBox:~$ g++ hw4.cpp -o hw4
tsai@tsai-VirtualBox:~$ ./hw4
0 1082-prog4-data.txt
1 0 3 9 9
2 0 4 10 10
3 0 5 15 15

0 t1: arrive
0 t2: arrive
0 t3: arrive
0 t1: start
3 t1: end
3 t2: start
7 t2: end
7 t3: start
9 t1: start
12 t1: end
12 t2: start
15 t3: deadline miss
```

基本功能(選項 1):

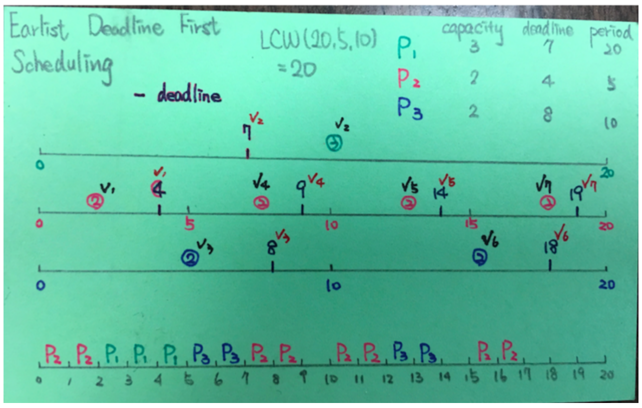


```
1 1082-prog4-data.txt
1 0 3 9 9
2 0 4 10 10
3 0 5 15 15

0 t1: arrive
0 t2: arrive
0 t3: arrive
0 t1: start
3 t1: end
3 t2: start
7 t2: end
7 t3: start
12 t3: end
12 t1: start
15 t1: end
15 t2: start
19 t2: end
19 t1: start
22 t1: end
22 t2: start
26 t2: end
26 t3: start
30 t3: deadline miss
```

進階功能(選項 2):



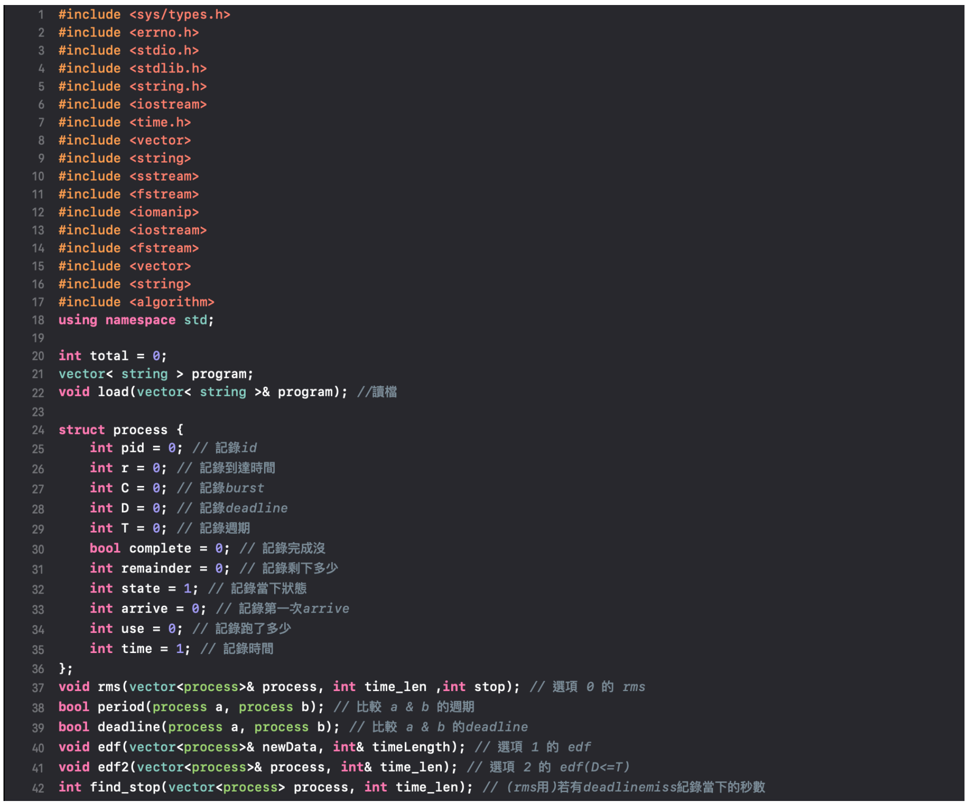
```
tsai@tsai-VirtualBox:~$ ./hw4
2 1082-prog4-data.txt
1 0 3 7 20
2 0 2 4 5
3 0 2 8 10

0 t1: arrive
0 t2: arrive
0 t3: arrive
0 t2: start
2 t2: end
2 t1: start
5 t1: end
5 t3: start
7 t3: end
7 t2: start
9 t2: end
10 t2: start
12 t2: end
12 t3: start
14 t3: end
15 t2: start
17 t2: end
20 t1: start
```



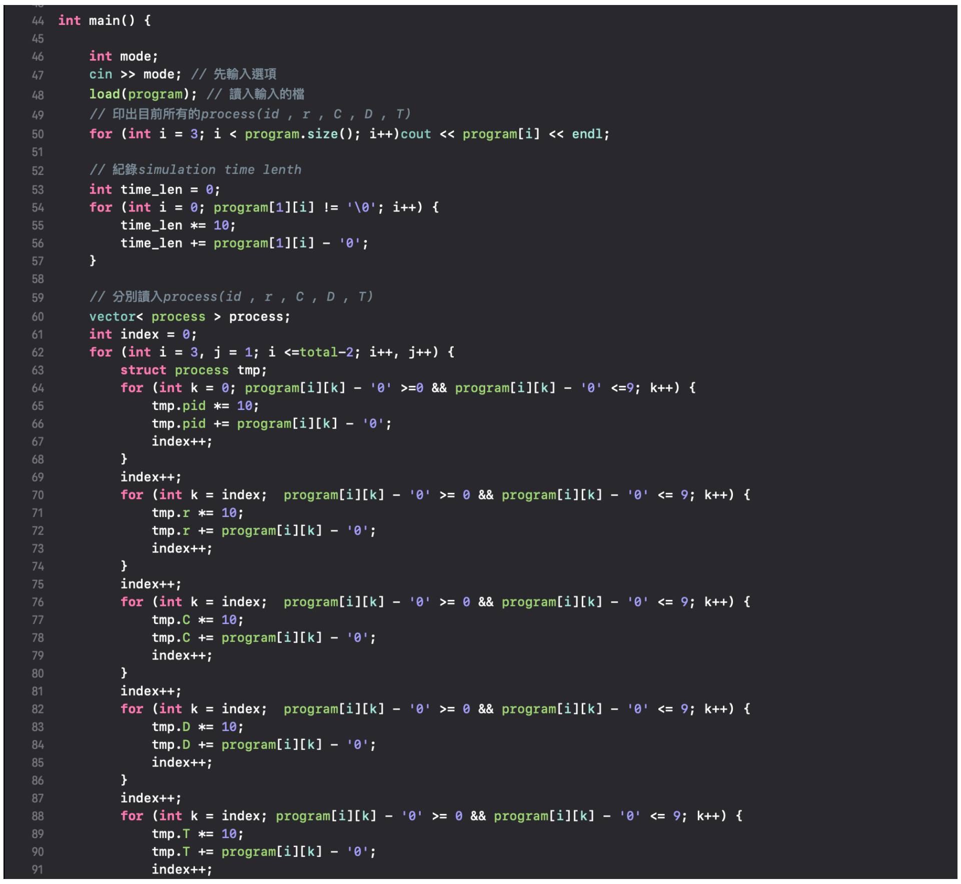
（符合右邊測資應出來的結果)

(右邊為參考測資)(D <= T)

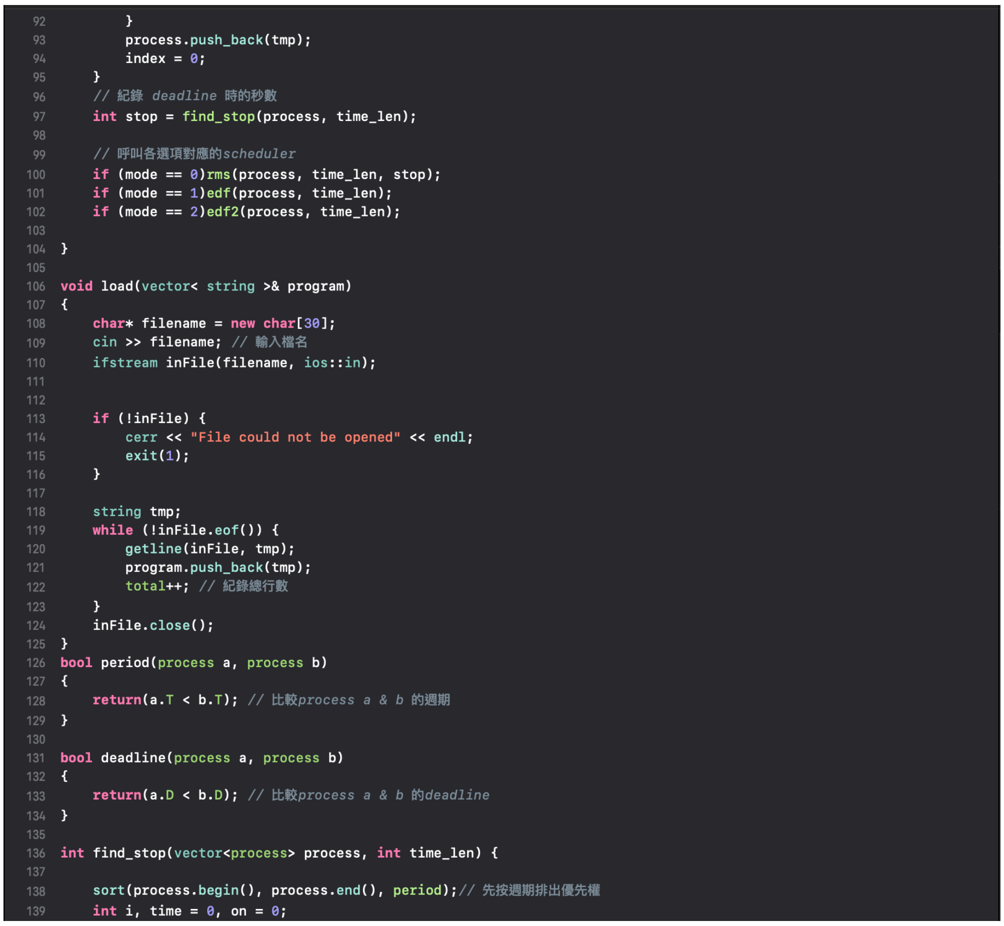
程式碼解釋：打在 //之後



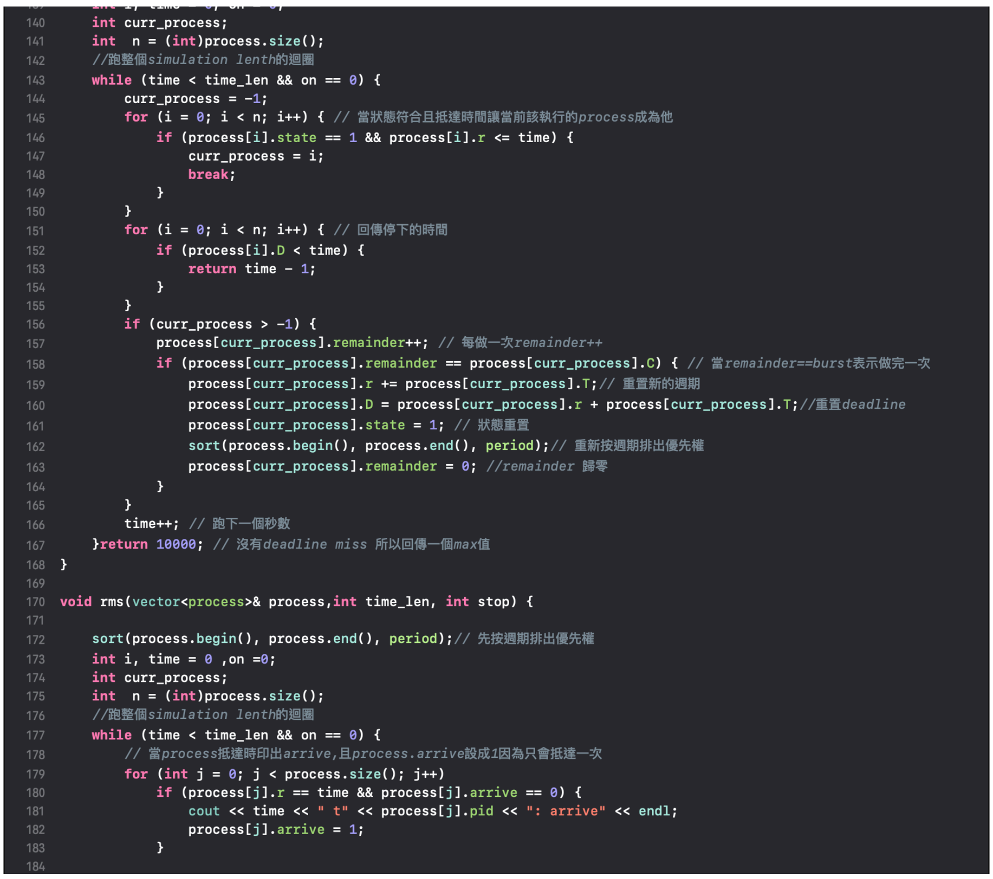
```cpp
#include <sys/types.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream>
#include <time.h>
#include <vector>
#include <string>
#include <sstream>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <algorithm>
using namespace std;

int total = 0;
vector< string > program;
void load(vector< string >& program); //讀檔

struct process {
    int pid = 0; // 記錄id
    int r = 0; // 記錄到達時間
    int C = 0; // 記錄burst
    int D = 0; // 記錄deadline
    int T = 0; // 記錄週期
    bool complete = 0; // 記錄完成沒
    int remainder = 0; // 記錄剩下多少
    int state = 1; // 記錄當下狀態
    int arrive = 0; // 記錄第一次arrive
    int use = 0; // 記錄跑了多少
    int time = 1; // 記錄時間
};
void rms(vector<process>& process, int time_len ,int stop); // 選項 0 的 rms
bool period(process a, process b); // 比較 a & b 的週期
bool deadline(process a, process b); // 比較 a & b 的deadline
void edf(vector<process>& newData, int& timeLength); // 選項 1 的 edf
void edf2(vector<process>& process, int& time_len); // 選項 2 的 edf(D<=T)
int find_stop(vector<process> process, int time_len); // (rms用)若有deadlinemiss紀錄當下的秒數
```



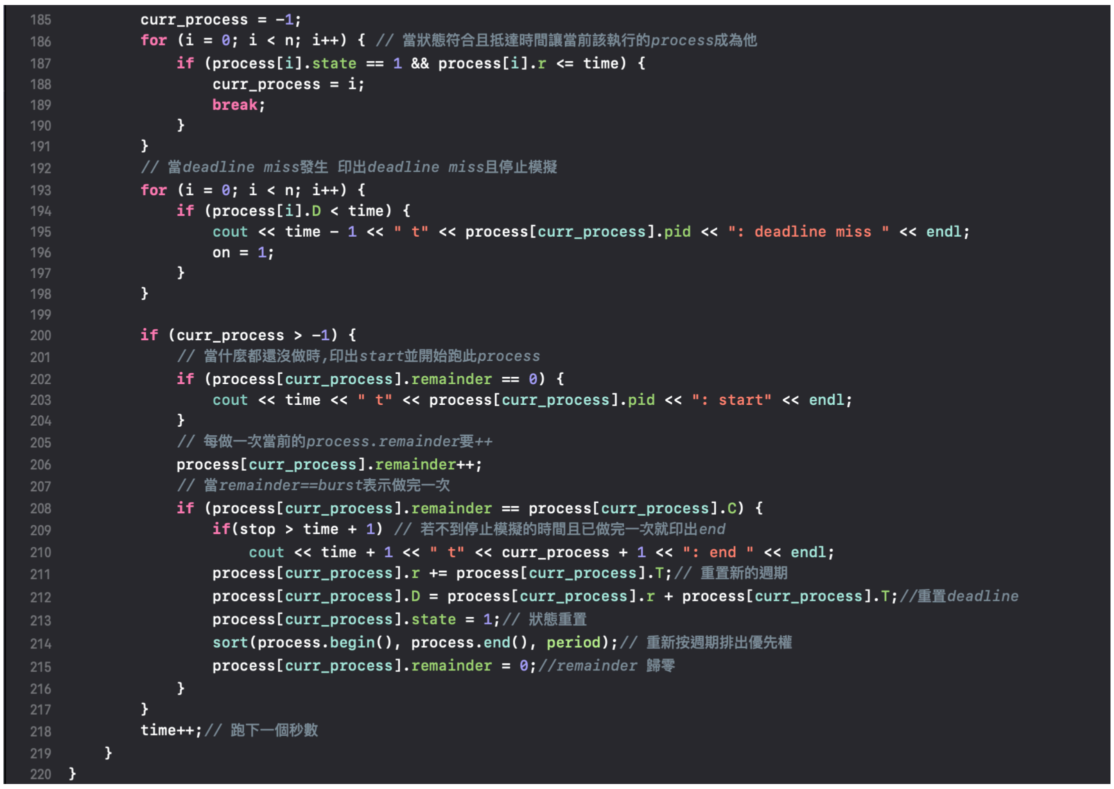
```cpp
int main() {

    int mode;
    cin >> mode; // 先輸入選項
    load(program); // 讀入輸入的檔
    // 印出目前所有的process(id , r , C , D , T)
    for (int i = 3; i < program.size(); i++)cout << program[i] << endl;

    // 紀錄simulation time lenth
    int time_len = 0;
    for (int i = 0; program[1][i] != '\0'; i++) {
        time_len *= 10;
        time_len += program[1][i] - '0';
    }

    // 分別讀入process(id , r , C , D , T)
    vector< process > process;
    int index = 0;
    for (int i = 3, j = 1; i <=total-2; i++, j++) {
        struct process tmp;
        for (int k = 0; program[i][k] - '0' >=0 && program[i][k] - '0' <=9; k++) {
            tmp.pid *= 10;
            tmp.pid += program[i][k] - '0';
            index++;
        }
        index++;
        for (int k = index;  program[i][k] - '0' >= 0 && program[i][k] - '0' <= 9; k++) {
            tmp.r *= 10;
            tmp.r += program[i][k] - '0';
            index++;
        }
        index++;
        for (int k = index;  program[i][k] - '0' >= 0 && program[i][k] - '0' <= 9; k++) {
            tmp.C *= 10;
            tmp.C += program[i][k] - '0';
            index++;
        }
        index++;
        for (int k = index;  program[i][k] - '0' >= 0 && program[i][k] - '0' <= 9; k++) {
            tmp.D *= 10;
            tmp.D += program[i][k] - '0';
            index++;
        }
        index++;
        for (int k = index; program[i][k] - '0' >= 0 && program[i][k] - '0' <= 9; k++) {
            tmp.T *= 10;
            tmp.T += program[i][k] - '0';
            index++;
```



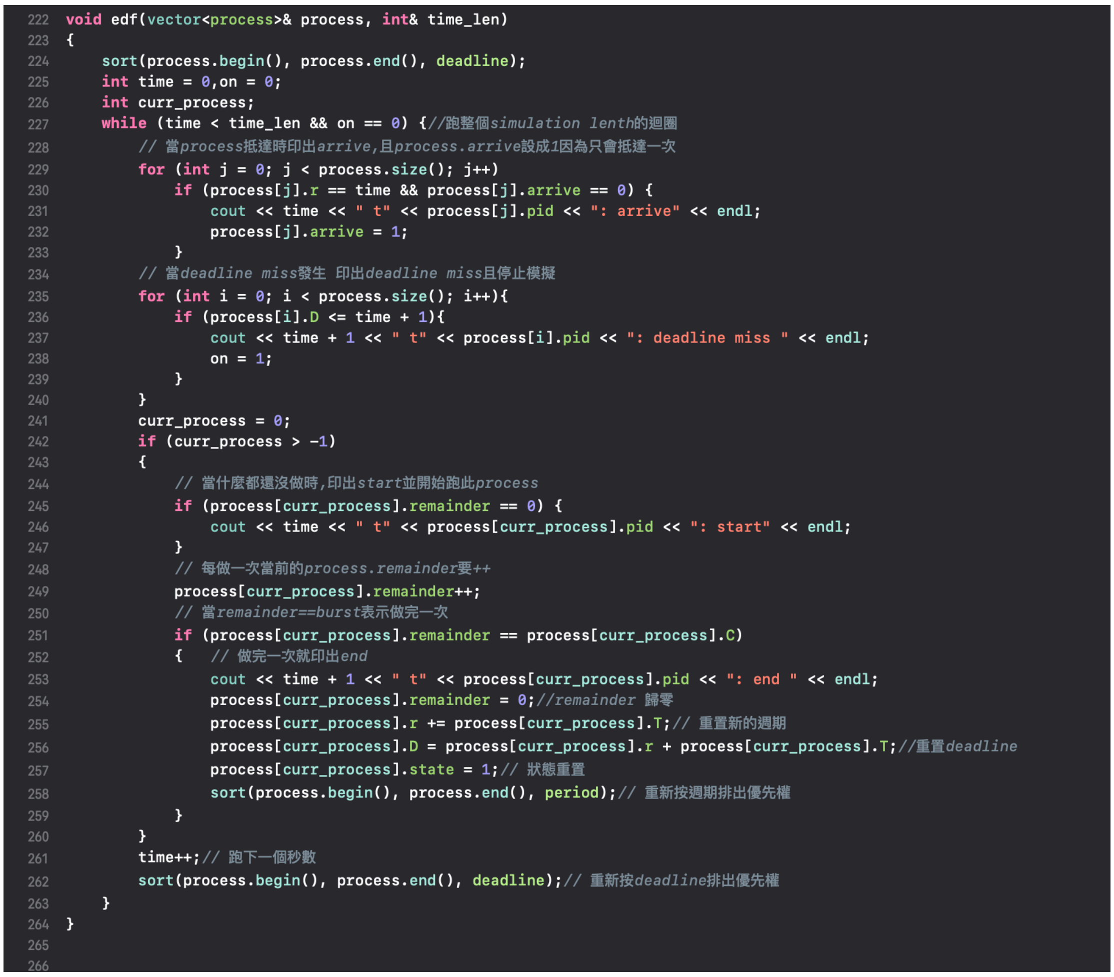
```cpp
        }
        process.push_back(tmp);
        index = 0;
    }
    // 紀錄 deadline 時的秒數
    int stop = find_stop(process, time_len);

    // 呼叫各選項對應的scheduler
    if (mode == 0)rms(process, time_len, stop);
    if (mode == 1)edf(process, time_len);
    if (mode == 2)edf2(process, time_len);

}

void load(vector< string >& program)
{
    char* filename = new char[30];
    cin >> filename; // 輸入檔名
    ifstream inFile(filename, ios::in);


    if (!inFile) {
        cerr << "File could not be opened" << endl;
        exit(1);
    }

    string tmp;
    while (!inFile.eof()) {
        getline(inFile, tmp);
        program.push_back(tmp);
        total++; // 紀錄總行數
    }
    inFile.close();
}
bool period(process a, process b)
{
    return(a.T < b.T); // 比較process a & b 的週期
}

bool deadline(process a, process b)
{
    return(a.D < b.D); // 比較process a & b 的deadline
}

int find_stop(vector<process> process, int time_len) {

    sort(process.begin(), process.end(), period);// 先按週期排出優先權
    int i, time = 0, on = 0;
```



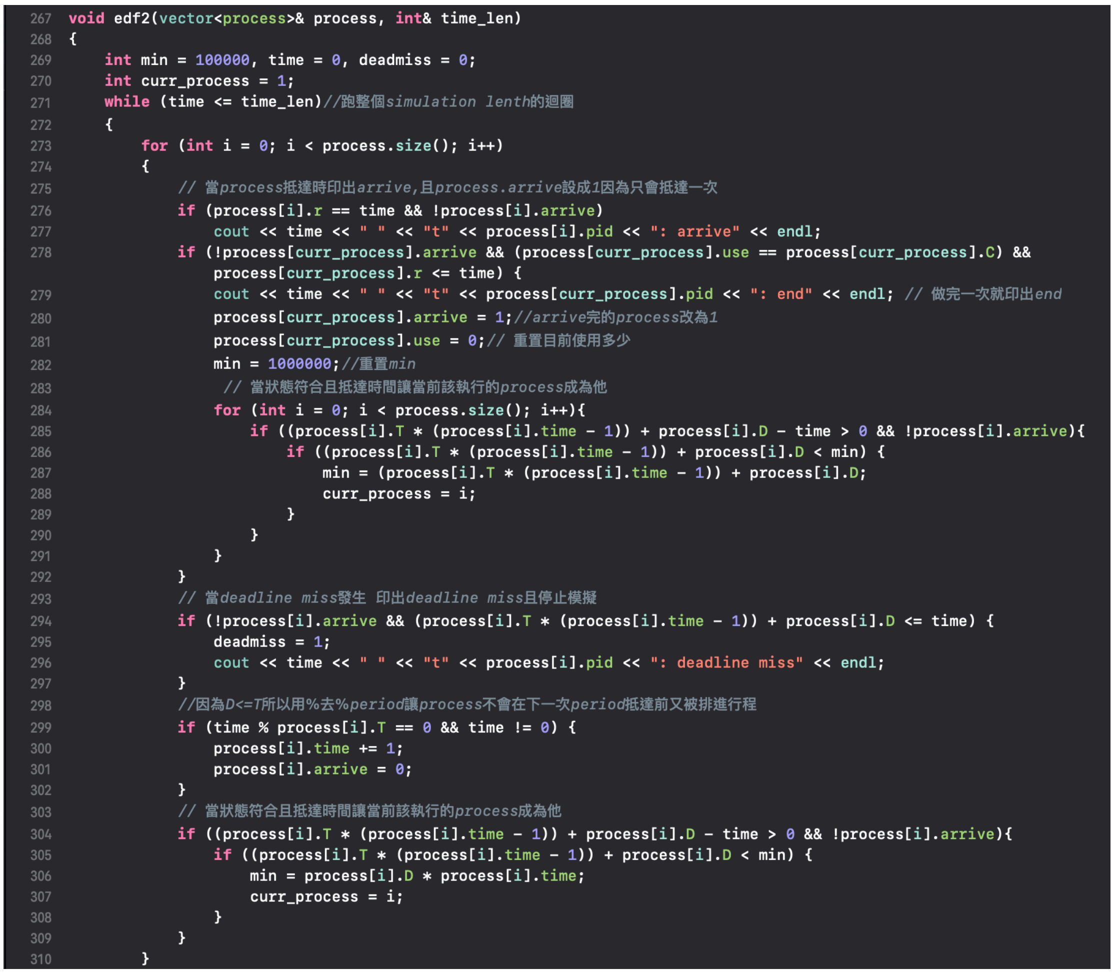
```cpp
    int curr_process;
    int  n = (int)process.size();
    //跑整個simulation lenth的迴圈
    while (time < time_len && on == 0) {
        curr_process = -1;
        for (i = 0; i < n; i++) { // 當狀態符合且抵達時間讓當前該執行的process成為他
            if (process[i].state == 1 && process[i].r <= time) {
                curr_process = i;
                break;
            }
        }
        for (i = 0; i < n; i++) { // 回傳停下的時間
            if (process[i].D < time) {
                return time - 1;
            }
        }
        if (curr_process > -1) {
            process[curr_process].remainder++; // 每做一次remainder++
            if (process[curr_process].remainder == process[curr_process].C) { // 當remainder==burst表示做完一次
                process[curr_process].r += process[curr_process].T;// 重置新的週期
                process[curr_process].D = process[curr_process].r + process[curr_process].T;//重置deadline
                process[curr_process].state = 1; // 狀態重置
                sort(process.begin(), process.end(), period);// 重新按週期排出優先權
                process[curr_process].remainder = 0; //remainder 歸零
            }
        }
        time++; // 跑下一個秒數
    }return 10000; // 沒有deadline miss 所以回傳一個max值
}

void rms(vector<process>& process,int time_len, int stop) {

    sort(process.begin(), process.end(), period);// 先按週期排出優先權
    int i, time = 0 ,on =0;
    int curr_process;
    int  n = (int)process.size();
    //跑整個simulation lenth的迴圈
    while (time < time_len && on == 0) {
        // 當process抵達時印出arrive,且process.arrive設成1因為只會抵達一次
        for (int j = 0; j < process.size(); j++)
            if (process[j].r == time && process[j].arrive == 0) {
                cout << time << " t" << process[j].pid << ": arrive" << endl;
                process[j].arrive = 1;
            }

```



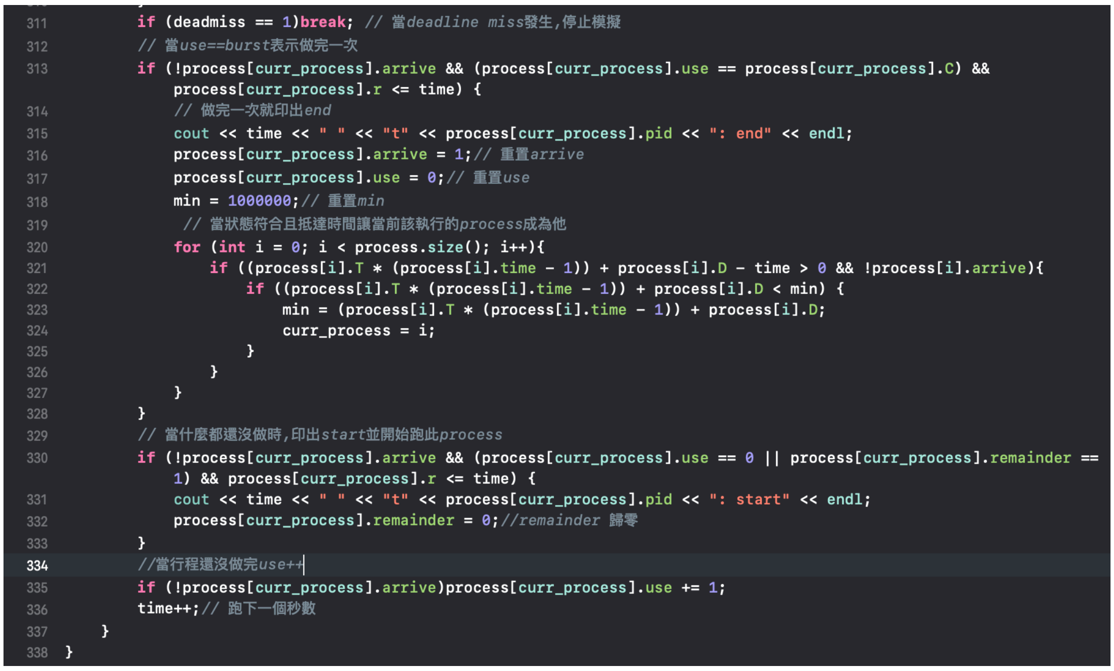
```cpp
        curr_process = -1;
        for (i = 0; i < n; i++) { // 當狀態符合且抵達時間讓當前該執行的process成為他
            if (process[i].state == 1 && process[i].r <= time) {
                curr_process = i;
                break;
            }
        }
        // 當deadline miss發生 印出deadline miss且停止模擬
        for (i = 0; i < n; i++) {
            if (process[i].D < time) {
                cout << time - 1 << " t" << process[curr_process].pid << ": deadline miss " << endl;
                on = 1;
            }
        }

        if (curr_process > -1) {
            // 當什麼都還沒做時,印出start並開始跑此process
            if (process[curr_process].remainder == 0) {
                cout << time << " t" << process[curr_process].pid << ": start" << endl;
            }
            // 每做一次當前的process.remainder要++
            process[curr_process].remainder++;
            // 當remainder==burst表示做完一次
            if (process[curr_process].remainder == process[curr_process].C) {
                if(stop > time + 1) // 若不到停止模擬的時間且已做完一次就印出end
                    cout << time + 1 << " t" << curr_process + 1 << ": end " << endl;
                process[curr_process].r += process[curr_process].T;// 重置新的週期
                process[curr_process].D = process[curr_process].r + process[curr_process].T;//重置deadline
                process[curr_process].state = 1;// 狀態重置
                sort(process.begin(), process.end(), period);// 重新按週期排出優先權
                process[curr_process].remainder = 0;//remainder 歸零
            }
        }
        time++;// 跑下一個秒數
    }
}
```



```cpp
void edf(vector<process>& process, int& time_len)
{
    sort(process.begin(), process.end(), deadline);
    int time = 0,on = 0;
    int curr_process;
    while (time < time_len && on == 0) {//跑整個simulation lenth的迴圈
        // 當process抵達時印出arrive,且process.arrive設成1因為只會抵達一次
        for (int j = 0; j < process.size(); j++)
            if (process[j].r == time && process[j].arrive == 0) {
                cout << time << " t" << process[j].pid << ": arrive" << endl;
                process[j].arrive = 1;
            }
        // 當deadline miss發生 印出deadline miss且停止模擬
        for (int i = 0; i < process.size(); i++){
            if (process[i].D <= time + 1){
                cout << time + 1 << " t" << process[i].pid << ": deadline miss " << endl;
                on = 1;
            }
        }
        curr_process = 0;
        if (curr_process > -1)
        {
            // 當什麼都還沒做時,印出start並開始跑此process
            if (process[curr_process].remainder == 0) {
                cout << time << " t" << process[curr_process].pid << ": start" << endl;
            }
            // 每做一次當前的process.remainder要++
            process[curr_process].remainder++;
            // 當remainder==burst表示做完一次
            if (process[curr_process].remainder == process[curr_process].C)
            {   // 做完一次就印出end
                cout << time + 1 << " t" << process[curr_process].pid << ": end " << endl;
                process[curr_process].remainder = 0;//remainder 歸零
                process[curr_process].r += process[curr_process].T;// 重置新的週期
                process[curr_process].D = process[curr_process].r + process[curr_process].T;//重置deadline
                process[curr_process].state = 1;// 狀態重置
                sort(process.begin(), process.end(), period);// 重新按週期排出優先權
            }
        }
        time++;// 跑下一個秒數
        sort(process.begin(), process.end(), deadline);// 重新按deadline排出優先權
    }
}
```



```cpp
void edf2(vector<process>& process, int& time_len)
{
    int min = 100000, time = 0, deadmiss = 0;
    int curr_process = 1;
    while (time <= time_len)//跑整個simulation lenth的迴圈
    {
        for (int i = 0; i < process.size(); i++)
        {
            // 當process抵達時印出arrive,且process.arrive設成1因為只會抵達一次
            if (process[i].r == time && !process[i].arrive)
                cout << time << " " << "t" << process[i].pid << ": arrive" << endl;
            if (!process[curr_process].arrive && (process[curr_process].use == process[curr_process].C) &&
                process[curr_process].r <= time) {
                cout << time << " " << "t" << process[curr_process].pid << ": end" << endl; // 做完一次就印出end
                process[curr_process].arrive = 1;//arrive完的process改為1
                process[curr_process].use = 0;// 重置目前使用多少
                min = 1000000;//重置min
                 // 當狀態符合且抵達時間讓當前該執行的process成為他
                for (int i = 0; i < process.size(); i++){
                    if ((process[i].T * (process[i].time - 1)) + process[i].D - time > 0 && !process[i].arrive){
                        if ((process[i].T * (process[i].time - 1)) + process[i].D < min) {
                            min = (process[i].T * (process[i].time - 1)) + process[i].D;
                            curr_process = i;
                        }
                    }
                }
            }
            // 當deadline miss發生 印出deadline miss且停止模擬
            if (!process[i].arrive && (process[i].T * (process[i].time - 1)) + process[i].D <= time) {
                deadmiss = 1;
                cout << time << " " << "t" << process[i].pid << ": deadline miss" << endl;
            }
            //因為D<=T所以用%去%period讓process不會在下一次period抵達前又被排進行程
            if (time % process[i].T == 0 && time != 0) {
                process[i].time += 1;
                process[i].arrive = 0;
            }
            // 當狀態符合且抵達時間讓當前該執行的process成為他
            if ((process[i].T * (process[i].time - 1)) + process[i].D - time > 0 && !process[i].arrive){
                if ((process[i].T * (process[i].time - 1)) + process[i].D < min) {
                    min = process[i].D * process[i].time;
                    curr_process = i;
                }
            }
        }
```



```cpp
        if (deadmiss == 1)break; // 當deadline miss發生,停止模擬
        // 當use==burst表示做完一次
        if (!process[curr_process].arrive && (process[curr_process].use == process[curr_process].C) &&
            process[curr_process].r <= time) {
            // 做完一次就印出end
            cout << time << " " << "t" << process[curr_process].pid << ": end" << endl;
            process[curr_process].arrive = 1;// 重置arrive
            process[curr_process].use = 0;// 重置use
            min = 1000000;// 重置min
             // 當狀態符合且抵達時間讓當前該執行的process成為他
            for (int i = 0; i < process.size(); i++){
                if ((process[i].T * (process[i].time - 1)) + process[i].D - time > 0 && !process[i].arrive){
                    if ((process[i].T * (process[i].time - 1)) + process[i].D < min) {
                        min = (process[i].T * (process[i].time - 1)) + process[i].D;
                        curr_process = i;
                    }
                }
            }
        }
        // 當什麼都還沒做時,印出start並開始跑此process
        if (!process[curr_process].arrive && (process[curr_process].use == 0 || process[curr_process].remainder ==
            1) && process[curr_process].r <= time) {
            cout << time << " " << "t" << process[curr_process].pid << ": start" << endl;
            process[curr_process].remainder = 0;//remainder 歸零
        }
        //當行程還沒做完use++
        if (!process[curr_process].arrive)process[curr_process].use += 1;
        time++;// 跑下一個秒數
    }
}
```