

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  $2 \leq n \leq 5$  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  $2 \leq n \leq 5$  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， $D \leq T$ 。模擬  $2 \leq n \leq 5$  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):

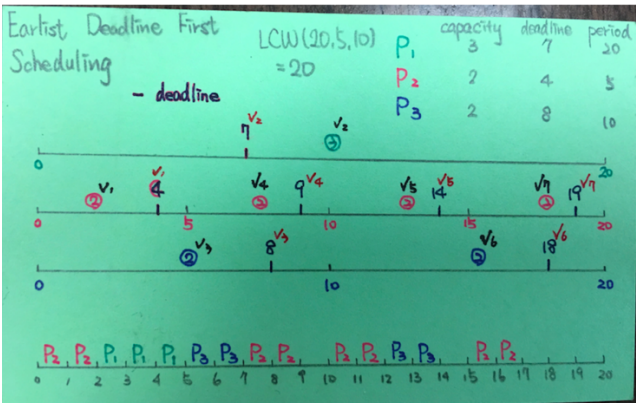
```
1 1082-prog4-data.txt
1 0 3 9 9
2 0 4 10 10
3 0 5 15 15

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) ( $D \leq T$ )

程式碼解釋: 打在 //之後

```
1  #include <sys/types.h>
2  #include <errno.h>
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <string.h>
6  #include <iostream>
7  #include <time.h>
8  #include <vector>
9  #include <string>
10 #include <sstream>
11 #include <fstream>
12 #include <iomanip>
13 #include <iostream>
14 #include <fstream>
15 #include <vector>
16 #include <string>
17 #include <algorithm>
18 using namespace std;
19
20 int total = 0;
21 vector< string > program;
22 void load(vector< string >& program); //讀檔
23
24 struct process {
25     int pid = 0; // 記錄id
26     int r = 0; // 記錄到達時間
27     int C = 0; // 記錄burst
28     int D = 0; // 記錄deadline
29     int T = 0; // 記錄週期
30     bool complete = 0; // 記錄完成沒
31     int remainder = 0; // 記錄剩下多少
32     int state = 1; // 記錄當下狀態
33     int arrive = 0; // 記錄第一次arrive
34     int use = 0; // 記錄跑了多少
35     int time = 1; // 記錄時間
36 };
37 void rms(vector<process>& process, int time_len, int stop); // 選項 0 的 rms
38 bool period(process a, process b); // 比較 a & b 的週期
39 bool deadline(process a, process b); // 比較 a & b 的deadline
40 void edf(vector<process>& newData, int& timeLength); // 選項 1 的 edf
41 void edf2(vector<process>& process, int& time_len); // 選項 2 的 edf(D<=T)
42 int find_stop(vector<process> process, int time_len); // (rms用)若有deadlinemiss記錄當下的秒數
```

```
44 int main() {
45
46     int mode;
47     cin >> mode; // 先輸入選項
48     load(program); // 讀入輸入的檔
49     // 印出目前所有的process(id , r , C , D , T)
50     for (int i = 3; i < program.size(); i++)cout << program[i] << endl;
51
52     // 記錄simulation time lenh
53     int time_len = 0;
54     for (int i = 0; program[i][i] != '\0'; i++) {
55         time_len += 10;
56         time_len += program[i][i] - '0';
57     }
58
59     // 分別讀入process(id , r , C , D , T)
60     vector< process > process;
61     int index = 0;
62     for (int i = 3, j = 1; i <= total-2; i++, j++) {
63         struct process tmp;
64         for (int k = 0; program[i][k] - '0' >= 0 && program[i][k] - '0' <= 9; k++) {
65             tmp.pid += 10;
66             tmp.pid += program[i][k] - '0';
67             index++;
68         }
69         index++;
70         for (int k = index; program[i][k] - '0' >= 0 && program[i][k] - '0' <= 9; k++) {
71             tmp.r += 10;
72             tmp.r += program[i][k] - '0';
73             index++;
74         }
75         index++;
76         for (int k = index; program[i][k] - '0' >= 0 && program[i][k] - '0' <= 9; k++) {
77             tmp.C += 10;
78             tmp.C += program[i][k] - '0';
79             index++;
80         }
81         index++;
82         for (int k = index; program[i][k] - '0' >= 0 && program[i][k] - '0' <= 9; k++) {
83             tmp.D += 10;
84             tmp.D += program[i][k] - '0';
85             index++;
86         }
87         index++;
88         for (int k = index; program[i][k] - '0' >= 0 && program[i][k] - '0' <= 9; k++) {
89             tmp.T += 10;
90             tmp.T += program[i][k] - '0';
91             index++;
92         }
93     }
94 }
```

```

92     }
93     process.push_back(tmp);
94     index = 0;
95 }
96 // 紀錄 deadline 時的秒數
97 int stop = find_stop(process, time_len);
98
99 // 呼叫各項對應的scheduler
100 if (mode == 0) rms(process, time_len, stop);
101 if (mode == 1) edf(process, time_len);
102 if (mode == 2) edf2(process, time_len);
103
104 }
105
106 void load(vector< string >& program)
107 {
108     char* filename = new char[30];
109     cin >> filename; // 輸入檔名
110     ifstream inFile(filename, ios::in);
111
112
113     if (!inFile) {
114         cerr << "File could not be opened" << endl;
115         exit(1);
116     }
117
118     string tmp;
119     while (!inFile.eof()) {
120         getline(inFile, tmp);
121         program.push_back(tmp);
122         total++; // 紀錄總行數
123     }
124     inFile.close();
125 }
126 bool period(process a, process b)
127 {
128     return(a.T < b.T); // 比較process a & b 的週期
129 }
130
131 bool deadline(process a, process b)
132 {
133     return(a.D < b.D); // 比較process a & b 的deadline
134 }
135
136 int find_stop(vector<process> process, int time_len) {
137
138     sort(process.begin(), process.end(), period); // 先按週期排出優先權
139     int i, time = 0, on = 0;

```

```

140     int curr_process;
141     int n = (int)process.size();
142     //跑整個simulation lenth的迴圈
143     while (time < time_len && on == 0) {
144         curr_process = -1;
145         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
146             if (process[i].state == 1 && process[i].r <= time) {
147                 curr_process = i;
148                 break;
149             }
150         }
151         for (i = 0; i < n; i++) { // 回傳停下的時間
152             if (process[i].D < time) {
153                 return time - 1;
154             }
155         }
156         if (curr_process > -1) {
157             process[curr_process].remainder++; // 每做一remainder++
158             if (process[curr_process].remainder == process[curr_process].C) { // 當remainder==burst表示做完一次
159                 process[curr_process].r += process[curr_process].T; // 重置新的週期
160                 process[curr_process].D = process[curr_process].r + process[curr_process].T; //重置deadline
161                 process[curr_process].state = 1; // 狀態重置
162                 sort(process.begin(), process.end(), period); // 重新按週期排出優先權
163                 process[curr_process].remainder = 0; //remainder 歸零
164             }
165         }
166         time++; // 跑下一個秒數
167     }return 10000; // 沒有deadline miss 所以回傳一個max值
168 }
169
170 void rms(vector<process>& process, int time_len, int stop) {
171
172     sort(process.begin(), process.end(), period); // 先按週期排出優先權
173     int i, time = 0, on = 0;
174     int curr_process;
175     int n = (int)process.size();
176     //跑整個simulation lenth的迴圈
177     while (time < time_len && on == 0) {
178         // 當process抵達時印出arrive,且process.arrive設成1因為只會抵達一次
179         for (int j = 0; j < process.size(); j++)
180             if (process[j].r == time && process[j].arrive == 0) {
181                 cout << time << " t" << process[j].pid << ": arrive" << endl;
182                 process[j].arrive = 1;
183             }
184

```

```

185     curr_process = -1;
186     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
187         if (process[i].state == 1 && process[i].r <= time) {
188             curr_process = i;
189             break;
190         }
191     }
192     // 當deadline miss發生 印出deadline miss且停止模擬
193     for (i = 0; i < n; i++) {
194         if (process[i].D < time) {
195             cout << time - 1 << " t" << process[curr_process].pid << ": deadline miss " << endl;
196             on = 1;
197         }
198     }
199
200     if (curr_process > -1) {
201         // 當什麼都還沒做時,印出start並開始跑此process
202         if (process[curr_process].remainder == 0) {
203             cout << time << " t" << process[curr_process].pid << ": start" << endl;
204         }
205         // 每做一次當前的process.remainder要++
206         process[curr_process].remainder++;
207         // 當remainder==burst表示做完一次
208         if (process[curr_process].remainder == process[curr_process].C) {
209             if (stop > time + 1) // 若不到停止模擬的時間且已做完一次就印出end
210                 cout << time + 1 << " t" << curr_process + 1 << ": end " << endl;
211             process[curr_process].r += process[curr_process].T; // 重置新的週期
212             process[curr_process].D = process[curr_process].r + process[curr_process].T; // 重置deadline
213             process[curr_process].state = 1; // 狀態重置
214             sort(process.begin(), process.end(), period); // 重新按週期排出優先權
215             process[curr_process].remainder = 0; // remainder 歸零
216         }
217     }
218     time++; // 跑下一個秒數
219 }
220 }

```

```

222 void edf(vector<process>& process, int& time_len)
223 {
224     sort(process.begin(), process.end(), deadline);
225     int time = 0, on = 0;
226     int curr_process;
227     while (time < time_len && on == 0) { // 跑整個simulation length的迴圈
228         // 當process抵達時印出arrive,且process.arrive設成1因為只會抵達一次
229         for (int j = 0; j < process.size(); j++)
230             if (process[j].r == time && process[j].arrive == 0) {
231                 cout << time << " t" << process[j].pid << ": arrive" << endl;
232                 process[j].arrive = 1;
233             }
234         // 當deadline miss發生 印出deadline miss且停止模擬
235         for (int i = 0; i < process.size(); i++) {
236             if (process[i].D <= time + 1) {
237                 cout << time + 1 << " t" << process[i].pid << ": deadline miss " << endl;
238                 on = 1;
239             }
240         }
241         curr_process = 0;
242         if (curr_process > -1)
243         {
244             // 當什麼都還沒做時,印出start並開始跑此process
245             if (process[curr_process].remainder == 0) {
246                 cout << time << " t" << process[curr_process].pid << ": start" << endl;
247             }
248             // 每做一次當前的process.remainder要++
249             process[curr_process].remainder++;
250             // 當remainder==burst表示做完一次
251             if (process[curr_process].remainder == process[curr_process].C)
252             {
253                 // 做完一次就印出end
254                 cout << time + 1 << " t" << process[curr_process].pid << ": end " << endl;
255                 process[curr_process].remainder = 0; // remainder 歸零
256                 process[curr_process].r += process[curr_process].T; // 重置新的週期
257                 process[curr_process].D = process[curr_process].r + process[curr_process].T; // 重置deadline
258                 process[curr_process].state = 1; // 狀態重置
259                 sort(process.begin(), process.end(), period); // 重新按週期排出優先權
260             }
261             time++; // 跑下一個秒數
262             sort(process.begin(), process.end(), deadline); // 重新按deadline排出優先權
263         }
264     }
265 }
266

```

```

267 void edf2(vector<process>& process, int& time_len)
268 {
269     int min = 100000, time = 0, deadmiss = 0;
270     int curr_process = 1;
271     while (time <= time_len) // 跑整個simulation length的迴圈
272     {
273         for (int i = 0; i < process.size(); i++)
274         {
275             // 當process抵達時印出arrive,且process.arrive設成1因為只會抵達一次
276             if (process[i].r == time && !process[i].arrive)
277                 cout << time << " " << "t" << process[i].pid << ": arrive" << endl;
278             if (!process[curr_process].arrive && (process[curr_process].use == process[curr_process].C) &&
                process[curr_process].r <= time) {
279                 cout << time << " " << "t" << process[curr_process].pid << ": end" << endl; // 做完一次就印出end
280                 process[curr_process].arrive = 1; // arrive完的process改為1
281                 process[curr_process].use = 0; // 重置目前使用多少
282                 min = 1000000; // 重置min
283                 // 當狀態符合且抵達時間讓當前該執行的process成為他
284                 for (int i = 0; i < process.size(); i++){
285                     if ((process[i].T * (process[i].time - 1)) + process[i].D - time > 0 && !process[i].arrive){
286                         if ((process[i].T * (process[i].time - 1)) + process[i].D < min) {
287                             min = (process[i].T * (process[i].time - 1)) + process[i].D;
288                             curr_process = i;
289                         }
290                     }
291                 }
292             }
293             // 當deadline miss發生 印出deadline miss且停止模擬
294             if (!process[i].arrive && (process[i].T * (process[i].time - 1)) + process[i].D <= time) {
295                 deadmiss = 1;
296                 cout << time << " " << "t" << process[i].pid << ": deadline miss" << endl;
297             }
298             // 因為D<=T所以用%去%period讓process不會在下次period抵達前又被排進行程
299             if (time % process[i].T == 0 && time != 0) {
300                 process[i].time += 1;
301                 process[i].arrive = 0;
302             }
303             // 當狀態符合且抵達時間讓當前該執行的process成為他
304             if ((process[i].T * (process[i].time - 1)) + process[i].D - time > 0 && !process[i].arrive){
305                 if ((process[i].T * (process[i].time - 1)) + process[i].D < min) {
306                     min = process[i].D * process[i].time;
307                     curr_process = i;
308                 }
309             }
310         }

```

```

311     if (deadmiss == 1) break; // 當deadline miss發生,停止模擬
312     // 當use==burst表示做完一次
313     if (!process[curr_process].arrive && (process[curr_process].use == process[curr_process].C) &&
        process[curr_process].r <= time) {
314         // 做完一次就印出end
315         cout << time << " " << "t" << process[curr_process].pid << ": end" << endl;
316         process[curr_process].arrive = 1; // 重置arrive
317         process[curr_process].use = 0; // 重置use
318         min = 1000000; // 重置min
319         // 當狀態符合且抵達時間讓當前該執行的process成為他
320         for (int i = 0; i < process.size(); i++){
321             if ((process[i].T * (process[i].time - 1)) + process[i].D - time > 0 && !process[i].arrive){
322                 if ((process[i].T * (process[i].time - 1)) + process[i].D < min) {
323                     min = (process[i].T * (process[i].time - 1)) + process[i].D;
324                     curr_process = i;
325                 }
326             }
327         }
328     }
329     // 當什麼都沒做時,印出start並開始跑此process
330     if (!process[curr_process].arrive && (process[curr_process].use == 0 || process[curr_process].remainder ==
        1) && process[curr_process].r <= time) {
331         cout << time << " " << "t" << process[curr_process].pid << ": start" << endl;
332         process[curr_process].remainder = 0; // remainder 歸零
333     }
334     // 當行程還沒做完use++
335     if (!process[curr_process].arrive) process[curr_process].use += 1;
336     time++; // 跑下一個秒數
337 }
338 }

```