10520104 電資四 莊東翰

1.開發環境:

Windows10 64位元 家用版

開發平台:

DEV C++

TDM-GCC 4.9.2 64-bit

程式語言:C++

2.資料結構:

(a) typedef 一個自訂型別名為 TaskType，其中以每個 process 為單位，紀錄所有有關此 process 的資訊。

```
struct TaskType{
  int pid;           // 紀錄 pid
  int cpu_burst;     // 紀錄 input 給予的 cpu burst
  int arrival;       // 紀錄 input 給予的造訪時間
  int priority;      // 紀錄 input 給予的優先權
  int remaining;     // 紀錄此 process 剩餘的 cpu burst (一開始與 cpu_burst 值相同)
  float response_ratio; // 紀錄此 process 的 response ratio
  float turnaround_time; // 紀錄此 process 的 turnaround time
  float waiting_time; // 紀錄此 process 的 waiting time
};
```

(b) 定義一個 vector 全域變數，用來記錄所有 process 的資訊

```
vector<TaskType> jobs;
```

(c) 每個排程方法的 function 內都有這些區域變數，分別用來記錄目前 queue 的情形與甘特圖(以每秒為單位紀錄)。

```
vector<int> gant_chart;
vector<TaskType> queue;
```

3.實作方法:
(a)Function: sort(string method);
將工作的到來次序先排序好，此 function 會根據每種排程法，在每個排程法開始計算以前將工作排序好。每次取工作時先確認 timer 是否等於 arrival，一樣的話就從頭開始依序取出。

排序順序:
FCFS & RR & HRRN: 先看 arrival，一樣再依 pid 大小排序。
SRTF: 先看 arrival，一樣再依 cpu burst 排序，若又一樣再依 pid 排序。
PPRR: 先看 arrival，一樣再依 priority 大小排序。

(b)
每個排程法的每回合都以 1 秒為單位。

FCFS:
從排序好的 task 列表中依序取出 task(當 timer == arrival)放入 queue。從 queue 的最前面的 task 開始執行並且每一回合 remaining – 1(做一秒)，依序完成 task 直到所有工作都被做完。

RR:
從排序好的 task 列表中依序取出 task(當 timer == arrival)放入 queue。
從 queue 的最前面的 task 開始執行並且每次 remaining – 1(做一秒)，再依據一個 time slice 為幾秒重複做上一個步驟。
須注意以下幾點:
1.每一秒都要確認有沒有新的工作到來並加入到 queue 中
2.新來的工作在 queue 中的順序要排在這回合完成的工作的前面。
3.若 time slice 還未用完就完成工作，則提早下一個 time slice 的開始並執行下一個工作。

SRTF:
從排序好的 task 列表中依序取出 task(當 timer == arrival)，從 queue 的最前面的 task(remaining 最少)開始執行並且每一回合 remaining – 1(做一秒)。
須注意以下幾點:
1.每一秒都要確認有沒有新的工作到來並加入到 queue 中
2.每一秒都要把 queue 中 remaining 最少的工作排至最前面執行。

PPRR:
從排序好的 task 列表中依序取出 task(當 timer == arrival)，從 queue 的最前面的 task (priority 最高)開始執行並且每一回合 remaining - 1(做一秒)。
須注意以下幾點:
1.每一秒都要確認有沒有新的工作到來並加入到 queue 中
2.每一秒都要把 queue 中 priority 最高的工作排至最前面執行。
3.若有多個工作都有最高的 priority，則套用 RR 的方式處理。
4.每回合 queue 對 priority 做排序時，不能動到幾個擁有相同 priority 工作的順序，因為那紀錄了之前 RR 的順序。

HRRN:
從排序好的 task 列表中依序取出 task(當 timer == arrival)，從 queue 的最前面的 task(response ratio 最高)開始執行並且每一回合 remaining - 1(做一秒)。
須注意以下幾點:
1.每一秒都要確認有沒有新的工作到來並加入到 queue 中
2.每一秒都要算 response ratio。