1-設計：

為使結構更清楚明瞭，將程式分為以下部分：

main.c： 主程式

process.c process.h：與 process 有關的變數以及函式

scheduler.c scheduler.h：與 scheduler 有關的變數以及函式

policy.c policy.h：實作 scheduling policy 的函式

本程式在雙核心系統上執行。主程式完成輸入後，會使用 fork 的方式，依據各個 process 的 ready time 生成 child process 來模擬，另外使用 sched_setaffinity 讓 scheduler (parent process) 與 process (child process) 使用不同的 CPU 以避免 scheduler 與 process 互相影響到彼此的執行時間 ，以 sched_setscheduler 改變 process 的 priority，將執行的 process 設為 SCHED_FIFO，等待的 process 設為 SCHED_IDLE。

Scheduler 以無窮迴圈實作，在迴圈內呼叫 UNIT_T（單位時間），一次迴圈相當於一個單位時間

Scheduler 流程：

完成輸入後，呼叫函式 scheduling 進入無窮迴圈直到所有 process 執行完畢

在每一次迴圈中：

檢查執行中的 process 是否已完成，若已完成則呼叫 waitpid 回收 process

檢查在此時間點是否有 process 生成，若有則呼叫 proc_create 生成此 process

呼叫 proc_next 依照給定的 scheduling policy 決定是否進行 context switch

呼叫 UNIT_T

2-核心版本：ubuntu 16.04 linux-4.14.25

核心數：2

virtual machine：Parallel

3-結果：

單位時間的計算是以各個 process 裡的迴圈實作，由於在每一次迴圈內 scheduler 都必須管理各個 process 而 process 本身不需要做任何事情，因此 scheduler 與 process 之間的時間會有所差別，不過因為 UNIT_T 本身執行時間夠長，所以造成的影響不大。

由於在大部份時間都會有一個 process (執行中) 的 priority 高於 SCHED_IDLE (等待中) ，因此可以保證在大部分時間 priority 被設為 SCHED_IDLE 的 process 不會被執行到，而極小部分會被執行到的時候在於高 priority 的 process 執行完畢的到指定高 priority 給下一個將被執行的 process 的這段區間，但因為此區間與 UNIT_T 比較起來很小，所以影響並不大。