The system flow and the explanation of the process(functions). (45%)

1. 執行maun.c 中的int main (void){}
2. 建立task
3. 透過OSTimeSet(0); 將OSTime歸零
4. 執行OSStart();
5. 進入os\_core.c中的void OSStart (void){}
6. 如果OSRunning處於等待步驟命令時開始執行OS\_SchedNew();
7. 如果OSPrioCur==0 也就是還沒有開始執行，

則列印"Tick\t Form Task\t To Task\n"以及OSTimeGet(), "\*\*\*\*\*\*\*\*", OSPrioHighRdy

否則，則列印OSTimeGet(), OSPrioCur, OSPrioHighRdy

1. 執行完OS\_SchedNew(); 之後更改 OSPrioCur、OSTCBHighRdy、OSTCBCur三個變數
2. 執行OSStartHighRdy ();
3. 執行context switch

(void task1(void\* p\_arg) {} 🡨🡪void task2(void\* p\_arg) {})

1. 循環執行6～10得以下循環

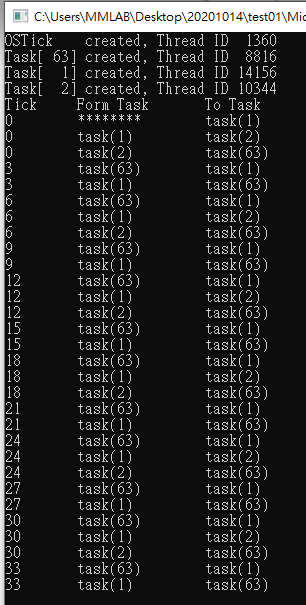
[task(1)🡪 task(2)🡪 task(63)🡪 task(1)🡪 task(63)]

當執行task(1)時得知task(2)優先權較高，則優先執行task(2)

當執行task(2)時得知task(63)優先權較高，則優先執行task(63)

當執行task(63)時得知task(1)優先權較高，則優先執行task(1)

The screenshot of the result. (10%)

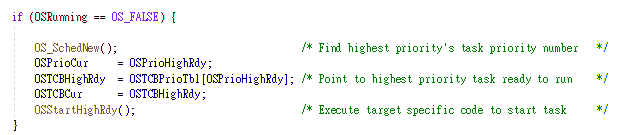


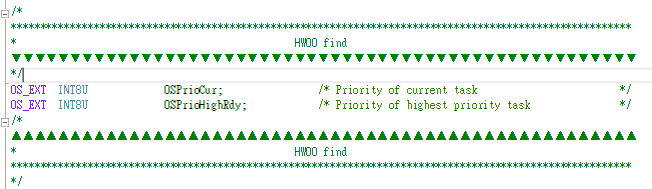
A report that describes your implementation (please attach the screenshot of the code and MARK the modified part). (45%)

1. 先從OSStart();的程式開始查看定義

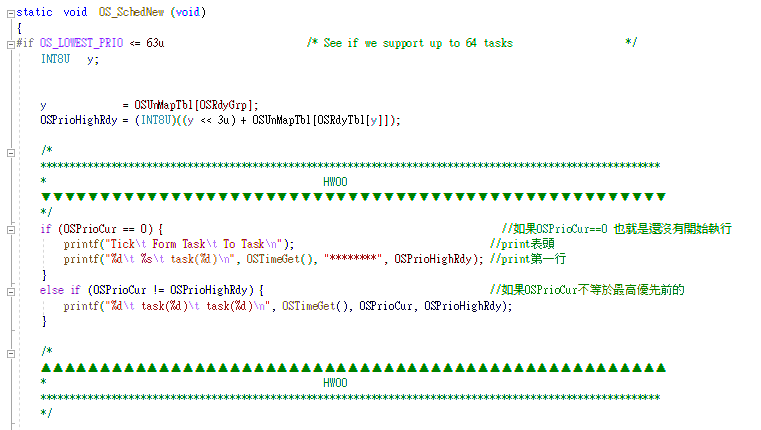


1. 在if (OSRunning == OS\_FALSE) {}中看見了本次作業需要的參數OSPrioCur與OSTCBHighRdy





1. 開始一行一行觀看if (OSRunning == OS\_FALSE) {}中函數的定義
2. 從static void OS\_SchedNew (void){}中可以發現OSTCBHighRdy的更新位置在這裡



1. 再來就是經由不斷嘗試得出最精簡的code以及符合題目要求的排版
2. END