|  |  |  |  |
| --- | --- | --- | --- |
| Embedded OS | PA1 | M11107309 | 何柏昇 |
| **PART I** | **Task Control Block Linked List** | | |
| Following the previous homework (HW1), please add some code to the μC/OS-II scheduler in the kernel level to observe the operations of the task control block (TCB) and TCB linked list. | | | |
| The screenshot results. (10%) | | | |
|  | | | |

|  |  |
| --- | --- |
| A report that describes your implementation (please attach the screenshot of the code and MARK the modified part). (10%) | |
| (如未標示，整段即為全新code) | |
| **os\_cpu\_c.c** | |
|  | |
| 建立task時會進行trace並輸出訊息，將其更改為題目要求的TCB address。 | |
| **os\_core.c** | |
| OSTCBInit() | |
|  | |
| OSStart() | |
|  | |
| 任務建立好後在OSStart，呼叫ptcb，輸出TCB list，利用ptcb->OSTCBNext與while，歷遍TCB list。 | |
| **PART II** | **RM Scheduler Implementation** |
| To implement the Rate Monotonic (RM) scheduler for periodic tasks and observe the scheduling behaviors. | |
| Periodic Task Set = {τID (ID, arrival time, execution time, period)}  Example Task Set 1 = {τ1 (1, 1, 2, 4), τ2 (2, 0, 4, 10)}  Example Task Set 2 = {τ1 (1, 3, 4, 14), τ2 (2, 0, 2, 8), τ3 (3, 0, 4, 10), τ4(4, 24, 2, 12)}  Example Task Set 3 = {τ1 (1, 2, 2, 10), τ2 (2, 1, 1, 5), τ3 (3, 0, 8, 15)} | |
| The correctness of schedule results of examples. Note the testing task set might not be the same as the given example task set. (40%) | |
| Task Set 1 | |
|  | |
| Task Set 2 | |
|  | |
| Task Set 3 | |
|  | |
| A report that describes your implementation (please attach the screenshot of the code and MARK the modified part). (40%) | |
| (如未標示，整段即為全新code) | |
| **ucos\_ii.h** | |
| global variable | |
|  | |
| 建立全域變數，用於後面計算response time。 | |

|  |
| --- |
| **main.c** |
| Main function |
|  |
| 依照題目要求，需要能夠根據輸入的任務數量，建立任務，所以利用for loop進行n個task的建立。  因為要進行RM scheduling，所以必須將task的priority根據period進行排列(成反比)，因此針對輸入的period將任務重新排列。 |

|  |
| --- |
| Task function |
|  |
| 根據給定的arrive time, execution time, period進行運作的task funtion。execution期間卡在while迴圈中不斷檢查執行時間是否結束,結束後則使用OSTimeDly()進入waiting狀態,直到下個周期開始。 |

|  |
| --- |
| **os\_core.c** |
| OSStart() |
|  |
| 根據題目要求，輸出標題。 |
| variable |
|  |
| 建立變數，等會用於計算response time、preemtion time、job number與CtxSw number。 |
| OSTimeTick() |
|  |
| 在Time tick中，設定task的啟動時間。 |

|  |
| --- |
| OS\_SchedNew() |
|  |
|  |
|  |
|  |
|  |
|  |
| 透過OSRdyTbl去確認task是否ready並進入waiting，來區分任務為completion與preemption，其中加入OSPrioCur與OSPrioHighRdy，可以協助判斷task的切換。 |

|  |  |
| --- | --- |
| 資料 | 判斷、記錄方式 |
| Event | Task a切換至Task b時,若 Task a 在 OSRdyTbl上仍處於ready狀態(==1)，則此情況為preemtpion；反之則為completion。 |
| Response Time | 「Completion的時間點」減掉「task 被設為ready的時間點(記錄於task\_start\_time)」，即為 response time。 |
| # of ContextSwitch | 記錄於task\_preempt\_count，每當task 切換則+1，completion輸出後歸零。 |
| Preemption Time | 發生 preemtpion時,記錄「task被preempt的時間點」,與「後續返回到該 task 的時間點」之差距,即為preemption time。再task complete之前有多次preemption則進行累加。 |
| **Report** | |
| 這次的PA1主要為RM Scheduling的實作，首先透過熟悉TCB的運作過程，再到如何計算Response Time等等工作時間，過程挫折感滿重的，需要將紙本的知識應用於實作上，常常因為不夠清楚，導致一直踩坑。  　　雖然沒有做出完整的功能，但卻更加熟悉RM的設計流程，非常有趣，也透過這次作業知道了生活上有許事情要安排，就好像這次作業一樣，要如何評估是非常重要的，才不會因為期中考與作業同時來的時候，導致Miss Deadline。 | |