

Project3

[PART I] NPCS Implementation

M10907324 吳俊逸

1.The screenshot result (with the given format) of the two task sets. (Time tick 0-100)

The NPCS of Output Result:

Task Set 1 = {task1 (2,5,30), task2 (3,3,60), task3 (0,7,90)}

```
D:\學校\台科\10901\EE5036701 嵌入式作業系統實作 Embedded OS Implementation\OS-code\μOSII_coc
OSTick created, Thread ID 11596
NPCS 第一題 Task Set 1 = {task1 (2,5,30), task2 (3,3,60), task3 (0,7,90)}

Tick    Event
0       Task3
1       Task3 get R2
6       Task3 release R2
6       Task1
7       Task1 get R1
10      Task1 release R1
11      Task2
14      Task3
15      Task63
32      Task1
33      Task1 get R1
36      Task1 release R1
37      Task63
62      Task1
63      Task1 get R1
66      Task1 release R1
66      Task2
69      Task1
70      Task63
90      Task3
91      Task3 get R2
96      Task3 release R2
96      Task1
97      Task1 get R1
100     Task1 release R1
101     Task3
102     Task63
```

Task Set 2 = {task1 (2,6,30), task2 (0,7,60)}

```
D:\學校\台科\10901\EE5036701 嵌入式作業系統實作 Embedded OS Implementat
OSTick created, Thread ID 16156
NPCS 第二題 Task Set 2 = {task1 (2,6,30), task2 (0,7,60)}

Tick    Event
0        Task2
1        Task2 get R2
4        Task2 release R2
4        Task2 get R1
6        Task2 release R1
6        Task1
7        Task1 get R1
9        Task1 release R1
9        Task1 get R2
11       Task1 release R2
12       Task2
13       Task63
32       Task1
33       Task1 get R1
35       Task1 release R1
35       Task1 get R2
37       Task1 release R2
38       Task63
60       Task2
61       Task2 get R2
64       Task2 release R2
64       Task2 get R1
66       Task2 release R1
66       Task1
67       Task1 get R1
69       Task1 release R1
69       Task1 get R2
71       Task1 release R2
72       Task2
73       Task63
92       Task1
93       Task1 get R1
95       Task1 release R1
95       Task1 get R2
97       Task1 release R2
98       Task63
120      Task2
121      Task2 release R2
```

2.A report that describes your implementation, including scheduling results of two task sets, modified functions, data structure, etc. (please ATTACH the screenshot of the code and MARK the modified part).

```

1998  *****
1999  *
2000  *
2001  */
2002  typedef struct {
2003      int arrivaTime; //到達時間
2004      int executionTime; //執行時間
2005      int periodTime; //週期時間
2006      int LR1_Time; //Lock R1 與到達時間的時間差
2007      int UR1_Time; //UnLock R1 與到達時間的時間差
2008      int LR2_Time; //Lock R2 與到達時間的時間差
2009      int UR2_Time; //UnLock R2 與到達時間的時間差
2010  }TaskData;
2011  /*
2012  *
2013  *
2014  *
2015  */

static TaskData Task1= {
    .arrivaTime=2,
    .executionTime=5,
    .periodTime=30,
    .LR1_Time=1,
    .UR1_Time=4,
    .LR2_Time=0,
    .UR2_Time=0
};

static TaskData Task2 = {
    .arrivaTime = 3,
    .executionTime = 3,
    .periodTime = 60,
    .LR1_Time = 0,
    .UR1_Time = 0,
    .LR2_Time = 0,
    .UR2_Time = 0
};

static TaskData Task3 = {
    .arrivaTime = 0,
    .executionTime = 7,
    .periodTime = 90,
    .LR1_Time = 0,
    .UR1_Time = 0,
    .LR2_Time = 1,
    .UR2_Time = 6
};

#define TASK_STACKSIZE 2048

#define TASK1_PRIORITY 2
#define TASK2_PRIORITY 4
#define TASK3_PRIORITY 5

#define TASK1_ID 1
#define TASK2_ID 2
#define TASK3_ID 3

static void task1(void* p_arg);
static void task2(void* p_arg);
static void task3(void* p_arg);

static OS_STK TASK1_STK[TASK_STACKSIZE];
static OS_STK TASK2_STK[TASK_STACKSIZE];
static OS_STK TASK3_STK[TASK_STACKSIZE];

```

```

57      *
58      *
59      */
60      void mywait(int tick)
61      {
62      #if OS_CRITICAL_METHOD==3
63          OS_CPU_SR cpu_sr = 0;
64      #endif
65          int now, exit;
66          OS_ENTER_CRITICAL();
67          now = OSTimeGet();
68          exit = now + tick;
69          OS_EXIT_CRITICAL();
70          while (1) {
71              if (exit <= OSTimeGet())
72                  break;
73          }
74      }
75      void OSTimeDly_arr(INT32U ticks, OS_TCB* task_123) //把arrival time tick 與 taskName 送進來
76      {
77          INT8U y;
78      #if OS_CRITICAL_METHOD == 3u /* Allocate storage for CPU status register */
79          OS_CPU_SR cpu_sr = 0u;
80      #endif
81          if (OSIntNesting > 0u) { /* See if trying to call from an ISR */
82              return;
83          }
84          if (OSLockNesting > 0u) { /* See if called with scheduler locked */
85              return;
86          }
87          if (ticks > 0u) { /* 0 means no delay! */
88              OS_ENTER_CRITICAL();
89              y = task_123->OSTCBY; /* Delay current task */
90              OSRdyTbl[y] &= (OS_PRIO)~task_123->OSTCBBitX;
91              OS_TRACE_TASK_SUSPENDED(task_123);
92              if (OSRdyTbl[y] == 0u) {
93                  OSRdyGrp &= (OS_PRIO)~task_123->OSTCBBitY;
94              }
95              task_123->OSTCBDly = ticks; /* Load ticks in TCB */
96              OS_TRACE_TASK_DLY(ticks);
97              OS_EXIT_CRITICAL();
98              //OS_Sched(); /* Find next task to run! */
99          }
100      }
101      /*
102      *
103      *
104      *
105      */

```

```

1755 /*
1756
1757 *
1758 *
1759 */
1760
1761 int t1 = 0; t2 = 0; t3 = 0;
1762 #define R1_PRIO 1
1763 #define R2_PRIO 3
1764 OS_EVENT* R1;
1765 OS_EVENT* R2;
1766 #include <windows.h>
1767 static void OS_SchedNew (void)
1768 {
1769     if (OS_LOWEST_PRIO <= 63u /* See if we support up to 64 tasks */
1770     {
1771         INT8U y;
1772         OSMapTbl[OSRdyGrp];
1773         OSHighRdy = (INT8U)(y << 3u) + OSMapTbl[OSRdyTbl[y]];
1774
1775         OS_TCB* ptcb;
1776         INT8U err;
1777         R1 = OSMutexCreate(R1_PRIO, &err);
1778         R2 = OSMutexCreate(R2_PRIO, &err);
1779
1780         if (OSPrrioCur > OSPrrioHighRdy) {
1781             ptcb = OSTCBBPrtbl[OSPrrioHighRdy];
1782             TaskData* point_t = ptcb->OSTCBExtPtr;
1783
1784             if (ptcb->OSTCBId == 65535) {
1785                 printf("M\t %s\n", OSTimeGet(), "Task63");//print 題目要求
1786                 t1 = 0; t2 = 0; t3 = 0;
1787             }
1788
1789             if (ptcb->OSTCBId == 1 && t1==0) {
1790                 t1 = 1;
1791                 printf("M\t %s\n", OSTimeGet(), "Task1");
1792                 if (point_t->Lr1_Time < point_t->Lr2_Time) { //如果這個Task是R1先做
1793                     if (point_t->Lr1_Time != 0 && point_t->Ur1_Time != 0) {
1794                         OSTimeDly_arr(point_t->Lr1_Time, OSTCBBPrtbl[2]);//讀Task等待get R1的時間
1795                         printf("M\t %s\n", OSTimeGet(), "Task1 get R1");//print 題目要求
1796                         OSMutexPend(R1, 0, &err);
1797                         OSTimeDly_arr(point_t->Ur1_Time - point_t->Lr1_Time, OSTCBBPrtbl[2]);//讀Task等待release R1的時間
1798                         printf("M\t %s\n", OSTimeGet(), "Task1 release R1");//print 題目要求
1799                         OSMutexPost(R1);
1800                         OSTimeDly_arr(point_t->arriveTime + point_t->executionTime - point_t->Ur1_Time, OSTCBBPrtbl[2]);//讀Task等待Task結束的時間
1801                     }
1802                     if (point_t->Lr2_Time != 0 && point_t->Ur2_Time != 0) {
1803                         OSTimeDly_arr(point_t->Lr1_Time, OSTCBBPrtbl[2]);//讀Task等待get R2的時間
1804                         printf("M\t %s\n", OSTimeGet(), "Task1 get R2");//print 題目要求
1805                         OSMutexPend(R2, 0, &err);
1806                         OSTimeDly_arr(point_t->arriveTime + point_t->executionTime - point_t->Ur2_Time, OSTCBBPrtbl[2]);//讀Task等待Task結束的時間
1807                     }
1808                 }
1809             }
1810         }
1811         else { //如果這個Task是R2先做
1812             if (point_t->Lr2_Time != 0 && point_t->Ur2_Time != 0) {
1813                 OSTimeDly_arr(point_t->Lr2_Time, OSTCBBPrtbl[2]);//讀Task等待get R2的時間
1814                 printf("M\t %s\n", OSTimeGet(), "Task2 get R2");//print 題目要求
1815                 OSMutexPend(R2, 0, &err);
1816                 OSTimeDly_arr(point_t->Ur2_Time - point_t->Lr2_Time, OSTCBBPrtbl[2]);//讀Task等待release R2的時間
1817                 printf("M\t %s\n", OSTimeGet(), "Task2 release R2");//print 題目要求
1818                 OSMutexPost(R2);
1819                 OSTimeDly_arr(point_t->arriveTime + point_t->executionTime - point_t->Ur2_Time, OSTCBBPrtbl[2]);//讀Task等待Task結束的時間
1820             }
1821             if (point_t->Lr1_Time != 0 && point_t->Ur1_Time != 0) {
1822                 OSTimeDly_arr(point_t->Lr1_Time, OSTCBBPrtbl[2]);//讀Task等待get R1的時間
1823                 printf("M\t %s\n", OSTimeGet(), "Task2 get R1");//print 題目要求
1824                 OSMutexPend(R1, 0, &err);
1825                 OSTimeDly_arr(point_t->Ur1_Time - point_t->Lr1_Time, OSTCBBPrtbl[2]);//讀Task等待release R1的時間
1826                 printf("M\t %s\n", OSTimeGet(), "Task2 release R1");//print 題目要求
1827                 OSMutexPost(R1);
1828                 OSTimeDly_arr(point_t->arriveTime + point_t->executionTime - point_t->Ur1_Time, OSTCBBPrtbl[2]);//讀Task等待Task結束的時間
1829             }
1830         }
1831     }
1832     if (ptcb->OSTCBId == 2 && t2 == 0) {
1833         t2 = 1;
1834         printf("M\t %s\n", OSTimeGet(), "Task2");//print 題目要求
1835
1836         if (point_t->Lr1_Time < point_t->Lr2_Time) { //如果這個Task是R1先做
1837             if (point_t->Lr1_Time != 0 && point_t->Ur1_Time != 0) {
1838                 OSTimeDly_arr(point_t->Lr1_Time, OSTCBBPrtbl[2]);//讀Task等待get R1的時間
1839                 printf("M\t %s\n", OSTimeGet(), "Task2 get R1");//print 題目要求
1840                 OSMutexPend(R1, 0, &err);
1841                 OSTimeDly_arr(point_t->Ur1_Time - point_t->Lr1_Time, OSTCBBPrtbl[2]);//讀Task等待release R1的時間
1842                 printf("M\t %s\n", OSTimeGet(), "Task2 release R1");//print 題目要求
1843                 OSMutexPost(R1);
1844                 OSTimeDly_arr(point_t->arriveTime + point_t->executionTime - point_t->Ur1_Time, OSTCBBPrtbl[2]);//讀Task等待Task結束的時間
1845             }
1846             if (point_t->Lr2_Time != 0 && point_t->Ur2_Time != 0) {
1847                 OSTimeDly_arr(point_t->Lr1_Time, OSTCBBPrtbl[2]);//讀Task等待get R2的時間
1848                 printf("M\t %s\n", OSTimeGet(), "Task2 get R2");//print 題目要求
1849             }
1850         }
1851         else { //如果這個Task是R2先做
1852             if (point_t->Lr2_Time != 0 && point_t->Ur2_Time != 0) {
1853                 OSTimeDly_arr(point_t->Lr2_Time, OSTCBBPrtbl[2]);//讀Task等待get R2的時間
1854                 printf("M\t %s\n", OSTimeGet(), "Task2 get R2");//print 題目要求
1855                 OSMutexPend(R2, 0, &err);
1856                 OSTimeDly_arr(point_t->Ur2_Time - point_t->Lr2_Time, OSTCBBPrtbl[2]);//讀Task等待release R2的時間
1857                 printf("M\t %s\n", OSTimeGet(), "Task2 release R2");//print 題目要求
1858                 OSMutexPost(R2);
1859                 OSTimeDly_arr(point_t->arriveTime + point_t->executionTime - point_t->Ur2_Time, OSTCBBPrtbl[2]);//讀Task等待Task結束的時間
1860             }
1861             if (point_t->Lr1_Time != 0 && point_t->Ur1_Time != 0) {
1862                 OSTimeDly_arr(point_t->Lr2_Time, OSTCBBPrtbl[2]);//讀Task等待get R1的時間
1863                 printf("M\t %s\n", OSTimeGet(), "Task2 get R1");//print 題目要求
1864                 OSMutexPend(R1, 0, &err);
1865                 OSTimeDly_arr(point_t->Ur1_Time - point_t->Lr2_Time, OSTCBBPrtbl[2]);//讀Task等待release R1的時間
1866                 printf("M\t %s\n", OSTimeGet(), "Task2 release R1");//print 題目要求
1867                 OSMutexPost(R1);
1868                 OSTimeDly_arr(point_t->arriveTime + point_t->executionTime - point_t->Ur1_Time, OSTCBBPrtbl[2]);//讀Task等待Task結束的時間
1869             }
1870         }
1871     }
1872 }
1873
1874 }
1875
1876 }
1877
1878 }
1879
1880 }
1881
1882 }
1883
1884 }
1885
1886 }
1887
1888 }
1889
1890 }
1891
1892 }
1893
1894 }
1895
1896 }
1897
1898 }
1899
1900 }
1901
1902 }
1903
1904 }
1905
1906 }
1907
1908 }
1909
1910 }
1911
1912 }
1913
1914 }
1915
1916 }
1917
1918 }
1919
1920 }
1921
1922 }
1923
1924 }
1925
1926 }
1927
1928 }
1929
1930 }
1931
1932 }
1933
1934 }
1935
1936 }
1937
1938 }
1939
1940 }
1941
1942 }
1943
1944 }
1945
1946 }
1947
1948 }
1949
1950 }
1951
1952 }
1953
1954 }
1955
1956 }
1957
1958 }
1959
1960 }
1961
1962 }
1963
1964 }
1965
1966 }
1967
1968 }
1969
1970 }
1971
1972 }
1973
1974 }
1975
1976 }
1977
1978 }
1979
1980 }
1981
1982 }
1983
1984 }
1985
1986 }
1987
1988 }
1989
1990 }
1991
1992 }
1993
1994 }
1995
1996 }
1997
1998 }
1999
2000 }
2001
2002 }
2003
2004 }
2005
2006 }
2007
2008 }
2009
2010 }
2011
2012 }
2013
2014 }
2015
2016 }
2017
2018 }
2019
2020 }
2021
2022 }
2023
2024 }
2025
2026 }
2027
2028 }
2029
2030 }
2031
2032 }
2033
2034 }
2035
2036 }
2037
2038 }
2039
2040 }
2041
2042 }
2043
2044 }
2045
2046 }
2047
2048 }
2049
2050 }
2051
2052 }
2053
2054 }
2055
2056 }
2057
2058 }
2059
2060 }
2061
2062 }
2063
2064 }
2065
2066 }
2067
2068 }
2069
2070 }
2071
2072 }
2073
2074 }
2075
2076 }
2077
2078 }
2079
2080 }
2081
2082 }
2083
2084 }
2085
2086 }
2087
2088 }
2089
2090 }
2091
2092 }
2093
2094 }
2095
2096 }
2097
2098 }
2099
2100 }
2101
2102 }
2103
2104 }
2105
2106 }
2107
2108 }
2109
2110 }
2111
2112 }
2113
2114 }
2115
2116 }
2117
2118 }
2119
2120 }
2121
2122 }
2123
2124 }
2125
```

```

1847 if (point_t->Lx2_Time != 0 && point_t->Ux2_Time != 0) {
1848     OSTimeDly_arr(point_t->Lx1_Time, OSTCBPrioTbl[2]); //讀Task等待get R2的時間
1849     printf("M\t %s\n", OSTimeGet(), "Task2 get R2");//print 題目要求
1850     OSMutexPend(R2, 0, &err);
1851     OSTimeDly_arr(point_t->Ux1_Time - point_t->Lx1_Time, OSTCBPrioTbl[2]); //讀Task等待release R2的時間
1852     printf("M\t %s\n", OSTimeGet(), "Task2 release R2");//print 題目要求
1853     OSMutexPost(R2);
1854     OSTimeDly_arr(point_t->arrivalTime + point_t->executionTime - point_t->Ux2_Time, OSTCBPrioTbl[2]); //讀Task等待Task結束的時間
1855 }
1856
1857 else { //如果這個Task是R2先做
1858     if (point_t->Lx2_Time != 0 && point_t->Ux2_Time != 0) {
1859         OSTimeDly_arr(point_t->Lx1_Time, OSTCBPrioTbl[2]); //讀Task等待get R2的時間
1860         printf("M\t %s\n", OSTimeGet(), "Task2 get R2");//print 題目要求
1861         OSMutexPend(R2, 0, &err);
1862         OSTimeDly_arr(point_t->Ux1_Time - point_t->Lx1_Time, OSTCBPrioTbl[2]); //讀Task等待release R2的時間
1863         printf("M\t %s\n", OSTimeGet(), "Task2 release R2");//print 題目要求
1864         OSMutexPost(R2);
1865         OSTimeDly_arr(point_t->arrivalTime + point_t->executionTime - point_t->Ux2_Time, OSTCBPrioTbl[2]); //讀Task等待Task結束的時間
1866     }
1867     if (point_t->Lx1_Time != 0 && point_t->Ux1_Time != 0) {
1868         OSTimeDly_arr(point_t->Lx1_Time, OSTCBPrioTbl[2]); //讀Task等待get R1的時間
1869         printf("M\t %s\n", OSTimeGet(), "Task2 get R1");//print 題目要求
1870         OSMutexPend(R1, 0, &err);
1871         OSTimeDly_arr(point_t->Ux1_Time - point_t->Lx1_Time, OSTCBPrioTbl[2]); //讀Task等待release R1的時間
1872         printf("M\t %s\n", OSTimeGet(), "Task2 release R1");//print 題目要求
1873         OSMutexPost(R1);
1874         OSTimeDly_arr(point_t->arrivalTime + point_t->executionTime - point_t->Ux1_Time, OSTCBPrioTbl[2]); //讀Task等待Task結束的時間
1875     }
1876 }
1877
1878 if (ptcb->OSTCBId == 3 && t3 == 0) {
1879     t3 = 1;
1880     printf("M\t %s\n", OSTimeGet(), "Task3");//print 題目要求
1881
1882     if (point_t->Lx1_Time < point_t->Lx2_Time) { //如果這個Task是R1先做
1883         if (point_t->Lx1_Time != 0 && point_t->Ux1_Time != 0) {
1884             OSTimeDly_arr(point_t->Lx1_Time, OSTCBPrioTbl[2]); //讀Task等待get R1的時間
1885             printf("M\t %s\n", OSTimeGet(), "Task3 get R1");//print 題目要求
1886             OSMutexPend(R1, 0, &err);
1887             OSTimeDly_arr(point_t->Ux1_Time - point_t->Lx1_Time, OSTCBPrioTbl[2]); //讀Task等待release R1的時間
1888             printf("M\t %s\n", OSTimeGet(), "Task3 release R1");//print 題目要求
1889             OSMutexPost(R1);
1890             OSTimeDly_arr(point_t->arrivalTime + point_t->executionTime - point_t->Ux1_Time, OSTCBPrioTbl[2]); //讀Task等待Task結束的時間
1891         }
1892
1893         OSMutexPost(R1);
1894         OSTimeDly_arr(point_t->arrivalTime + point_t->executionTime - point_t->Ux1_Time, OSTCBPrioTbl[2]); //讀Task等待Task結束的時間
1895     }
1896     if (point_t->Lx2_Time != 0 && point_t->Ux2_Time != 0) {
1897         OSTimeDly_arr(point_t->Lx1_Time, OSTCBPrioTbl[2]); //讀Task等待get R2的時間
1898         printf("M\t %s\n", OSTimeGet(), "Task3 get R2");//print 題目要求
1899         OSMutexPend(R2, 0, &err);
1900         OSTimeDly_arr(point_t->Ux1_Time - point_t->Lx1_Time, OSTCBPrioTbl[2]); //讀Task等待release R2的時間
1901         printf("M\t %s\n", OSTimeGet(), "Task3 release R2");//print 題目要求
1902         OSMutexPost(R2);
1903         OSTimeDly_arr(point_t->arrivalTime + point_t->executionTime - point_t->Ux2_Time, OSTCBPrioTbl[2]); //讀Task等待Task結束的時間
1904     }
1905     else { //如果這個Task是R2先做
1906         if (point_t->Lx2_Time != 0 && point_t->Ux2_Time != 0) {
1907             OSTimeDly_arr(point_t->Lx1_Time, OSTCBPrioTbl[2]); //讀Task等待get R2的時間
1908             printf("M\t %s\n", OSTimeGet(), "Task3 get R2");//print 題目要求
1909             OSMutexPend(R2, 0, &err);
1910             OSTimeDly_arr(point_t->Ux1_Time - point_t->Lx1_Time, OSTCBPrioTbl[2]); //讀Task等待release R2的時間
1911             printf("M\t %s\n", OSTimeGet(), "Task3 release R2");//print 題目要求
1912             OSMutexPost(R2);
1913             OSTimeDly_arr(point_t->arrivalTime + point_t->executionTime - point_t->Ux2_Time, OSTCBPrioTbl[2]); //讀Task等待Task結束的時間
1914         }
1915         if (point_t->Lx1_Time != 0 && point_t->Ux1_Time != 0) {
1916             OSTimeDly_arr(point_t->Lx1_Time, OSTCBPrioTbl[2]); //讀Task等待get R1的時間
1917             printf("M\t %s\n", OSTimeGet(), "Task3 get R1");//print 題目要求
1918             OSMutexPend(R1, 0, &err);
1919             OSTimeDly_arr(point_t->Ux1_Time - point_t->Lx1_Time, OSTCBPrioTbl[2]); //讀Task等待release R1的時間
1920             printf("M\t %s\n", OSTimeGet(), "Task3 release R1");//print 題目要求
1921             OSMutexPost(R1);
1922             OSTimeDly_arr(point_t->arrivalTime + point_t->executionTime - point_t->Ux1_Time, OSTCBPrioTbl[2]); //讀Task等待Task結束的時間
1923         }
1924     }
1925     ptcb = ptcb->OSTCBNext;
1926 }
1927
1928 /*
1929 *****
1930 ***** PAGE3-NPCS *****
1931 *****
1932 */

```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242
--	---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

[PART II] CPP Implementation

1. The screenshot result (with the given format) of the two task sets. (Time tick 0- 100)

Task Set 1 = {task1 (2,5,30), task2 (3,3,60), task3 (0,7,90)}

```
D:\學校\台科\10901\EE5036701 嵌入式作業系統實作 Embedded OS Implementation\OS_code\μOSII
OSTick created, Thread ID 9192
CPP 第一題 Task Set 2 = {task1 (2,5,30), task2 (3,3,60), task3 (0,7,90)}
pro T1=2 T2=3 T3=5 R1=1 R2=4

Tick    Event                Prio_Inheritance
0        Task3
1        Task3 get R2         5->4
2        Task1
3        Task1 get R1         2->1
6        Task1 release R1     1->2
7        Task2
14       Task3 release R2     4->5
15       Task63
32       Task1
33       Task1 get R1         2->1
36       Task1 get R1         2->1
37       Task63
62       Task1
63       Task1 get R1         2->1
66       Task1 get R1         2->1
67       Task2
70       Task63
90       Task3
91       Task3 get R2         5->4
92       Task1
93       Task1 get R1         2->1
96       Task1 release R1     1->2
101      Task3 release R2     4->5
102      Task63
```

Task Set 2 = {task1 (2,6,30), task2 (0,7,60)}

```
選取 D:\學校\台科\10901\EE5036701 嵌入式作業系統實作 Embedded OS Impler
OSTick created, Thread ID 11788
CPP 第二題 Task Set 2 = {task1 (2,6,30), task2 (0,7,60)}
pro T1=3 T2=4 R1=1 R2=2

Tick    Event                Prio_Inheritance
0        Task2
1        Task2 get R2          4->2
4        Task2 release R2      2->1
4        Task2 get R1          1->1
6        Task2 release R1      1->1
6        Task3
7        Task1 get R1          3->1
9        Task1 release R1      1->1
9        Task1 get R2          1->2
11       Task1 release R2      2->3
12       Task2
13       Task63
32       Task1
33       Task1 get R1          3->1
35       Task1 release R1      1->1
35       Task1 get R2          1->2
37       Task1 release R2      2->3
38       Task63
60       Task2
61       Task2 get R2          4->2
64       Task2 release R2      2->1
64       Task2 get R1          1->1
66       Task2 release R1      1->1
66       Task1
67       Task1 get R1          3->1
69       Task1 release R1      1->1
69       Task1 get R2          1->2
71       Task1 release R2      2->3
72       Task2
73       Task63
92       Task1
93       Task1 get R1          3->1
95       Task1 release R1      1->1
95       Task1 get R2          1->2
97       Task1 release R2      2->3
98       Task63
120      Task2
121      Task2 get R2          4->2
```


2.A report that describes your implementation, including scheduling results of two task sets, modified functions, data structure, etc. (please ATTACH the screenshot of the code and MARK the modified part).

```

main.c+  os_time.c  ucos_ii.h  os_core.c+
OS2  (全域範圍)
1998  *****
1999  *
2000  *                                     PA03
2001  */
2002  typedef struct {
2003      int arrivaTime; //到達時間
2004      int executionTime; //執行時間
2005      int periodTime; //週期時間
2006      int LR1_Time; //Lock R1 與到達時間的時間差
2007      int UR1_Time; //UnLock R1 與到達時間的時間差
2008      int LR2_Time; //Lock R2 與到達時間的時間差
2009      int UR2_Time; //UnLock R2 與到達時間的時間差
2010  }TaskData;
2011  /*
2012  *
2013  *                                     PA03
2014  *
2015  */

```

```

static TaskData Task1= {
    .arrivaTime=2,
    .executionTime=5,
    .periodTime=30,
    .LR1_Time=1,
    .UR1_Time=4,
    .LR2_Time=0,
    .UR2_Time=0
};

static TaskData Task2 = {
    .arrivaTime = 3,
    .executionTime = 3,
    .periodTime = 60,
    .LR1_Time = 0,
    .UR1_Time = 0,
    .LR2_Time = 0,
    .UR2_Time = 0
};

static TaskData Task3 = {
    .arrivaTime = 0,
    .executionTime = 7,
    .periodTime = 90,
    .LR1_Time = 0,
    .UR1_Time = 0,
    .LR2_Time = 1,
    .UR2_Time = 6
};

```

```

#define TASK_STACKSIZE 2048
#define TASK1_PRIORITY 2
#define TASK2_PRIORITY 3
#define TASK3_PRIORITY 5
#define TASK1_ID 1
#define TASK2_ID 2
#define TASK3_ID 3
static void task1(void* p_arg);
static void task2(void* p_arg);
static void task3(void* p_arg);
static OS_STK TASK1_STK[TASK_STACKSIZE];
static OS_STK TASK2_STK[TASK_STACKSIZE];
static OS_STK TASK3_STK[TASK_STACKSIZE];

```

```

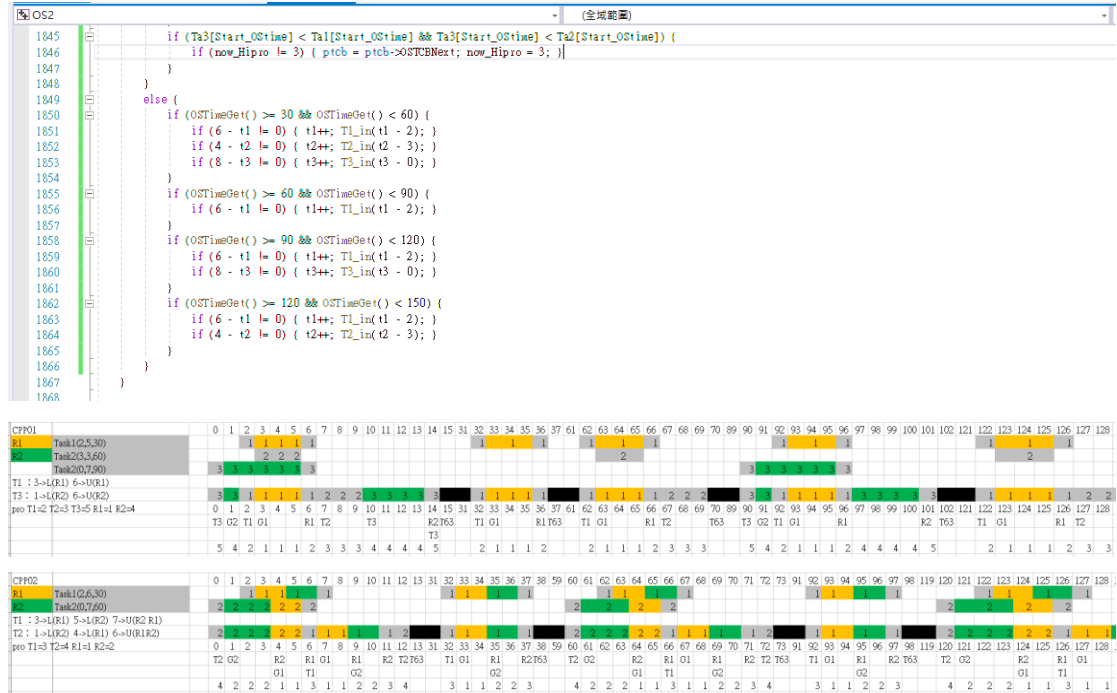
1755  /*
1756  *
1757  * PA03-NPCS
1758  */
1759
1760  int Ta1[] = { 63, 63, 2, 1, 1, 1, 2 };
1761  int Ta2[] = { 63, 63, 63, 3, 3, 3, 63 };
1762  int Ta3[] = { 5, 4, 4, 4, 4, 4, 5 };
1763
1764  int now_Hiprio = 0;
1765  int Start_OStime = 0;
1766  int t1 = 0; t2 = 0; t3 = 0;
1767  int a1 = 5; a2 = 3; a3 = 8;
1768
1769  void T1_in(int c) {
1770      OS_TCB* ptcb;
1771      ptcb = OSTCBPrioTbl[OSPrioHighRdy];
1772      TaskData* point_t = ptcb->OSTCBEExtPtr;
1773      if (c == 1) { printf("%d\t Task1\n", OSTimeGet()); } //print 题目要求
1774      if (c == point_t->LR1_Time + 1) { printf("%d\t Task1 get R1 %d->%d\n", OSTimeGet(), Ta1[Start_OStime - 1], Ta1[Start_OStime]); } //print 题目要求
1775      if (c == point_t->LR1_Time + 1) { printf("%d\t Task1 release R1 %d->%d\n", OSTimeGet(), Ta1[Start_OStime - 1], Ta1[Start_OStime]); } //print 题目要求
1776      if (c == point_t->LR2_Time + 1) { printf("%d\t Task1 get R2 %d->%d\n", OSTimeGet(), Ta1[Start_OStime - 1], Ta1[Start_OStime]); } //print 题目要求
1777      if (c == point_t->LR2_Time + 1) { printf("%d\t Task1 release R2 %d->%d\n", OSTimeGet(), Ta1[Start_OStime - 1], Ta1[Start_OStime]); } //print 题目要求
1778  }
1779
1780  void T2_in(int c) {
1781      OS_TCB* ptcb;
1782      ptcb = OSTCBPrioTbl[OSPrioHighRdy];
1783      TaskData* point_t = ptcb->OSTCBEExtPtr;
1784      if (c == 1) { printf("%d\t Task2\n", OSTimeGet()); } //print 题目要求
1785      if (c == point_t->LR1_Time + 1) { printf("%d\t Task2 get R1 %d->%d\n", OSTimeGet(), Ta2[Start_OStime - 1], Ta2[Start_OStime]); } //print 题目要求
1786      if (c == point_t->LR1_Time + 1) { printf("%d\t Task2 release R1 %d->%d\n", OSTimeGet(), Ta2[Start_OStime - 1], Ta2[Start_OStime]); } //print 题目要求
1787      if (c == point_t->LR2_Time + 1) { printf("%d\t Task2 get R2 %d->%d\n", OSTimeGet(), Ta2[Start_OStime - 1], Ta2[Start_OStime]); } //print 题目要求
1788      if (c == point_t->LR2_Time + 1) { printf("%d\t Task2 release R2 %d->%d\n", OSTimeGet(), Ta2[Start_OStime - 1], Ta2[Start_OStime]); } //print 题目要求
1789  }
1790
1791  void T3_in(int c) {
1792      OS_TCB* ptcb;
1793      ptcb = OSTCBPrioTbl[OSPrioHighRdy];
1794      TaskData* point_t = ptcb->OSTCBEExtPtr;
1795      if (c == 1) { printf("%d\t Task3\n", OSTimeGet()); } //print 题目要求
1796      if (c == point_t->LR1_Time + 1) { printf("%d\t Task3 get R1 %d->%d\n", OSTimeGet(), Ta3[Start_OStime - 1], Ta3[Start_OStime]); } //print 题目要求
1797      if (c == point_t->LR1_Time + 1) { printf("%d\t Task3 release R1 %d->%d\n", OSTimeGet(), Ta3[Start_OStime - 1], Ta3[Start_OStime]); } //print 题目要求
1798      if (c == point_t->LR2_Time + 1) { printf("%d\t Task3 get R2 %d->%d\n", OSTimeGet(), Ta3[Start_OStime - 1], Ta3[Start_OStime]); } //print 题目要求
1799      if (c == point_t->LR2_Time + 1) { printf("%d\t Task3 release R2 %d->%d\n", OSTimeGet(), Ta3[Start_OStime - 1], Ta3[Start_OStime]); } //print 题目要求
1800  }

```

```

1800
1801  #define R1_Prio 1
1802  #define R2_Prio 4
1803  OS_EVENT* R1;
1804  OS_EVENT* R2;
1805
1806  static void OS_SchedNew (void)
1807  {
1808      #if OS_LOWEST_Prio <= 63u /* See if we support up to 64 tasks */
1809      INT8U y;
1810      y = OSUnMapTbl[OSRdyGrp];
1811      OSPrioHighRdy = (INT8U)((y << 3u) + OSUnMapTbl[OSRdyTbl[y]]);
1812
1813      OS_TCB* ptcb;
1814      INT8U err;
1815      R1 = OSMutexCreate(R1_Prio, &err);
1816      R2 = OSMutexCreate(R2_Prio, &err);
1817      if (OSPrioCur != OSPrioHighRdy) {
1818          ptcb = OSTCBPrioTbl[OSPrioHighRdy];
1819          TaskData* point_t = ptcb->OSTCBEExtPtr;
1820          if (ptcb->OSTCBId == 65535) {
1821              printf("%d\t %s\n", OSTimeGet(), "Task63"); //print 题目要求
1822              t1 = 0; t2 = 0; t3 = 0;
1823          }
1824          if (ptcb->OSTCBId == 1) {
1825              if (OSTimeGet() < 30) { t1++; T1_in(t1); }
1826              else { t1++; T1_in(t1-2); }
1827          }
1828          if (ptcb->OSTCBId == 2) {
1829              if (OSTimeGet() < 30) { t2++; T2_in(t2); }
1830              else { t2++; T2_in(t2-2); }
1831          }
1832          if (ptcb->OSTCBId == 3) {
1833              if (OSTimeGet() < 30) { t3++; T3_in(t3); }
1834              else { t3++; T3_in(t3-2); }
1835          }
1836          Start_OStime = OSTimeGet() % 30;
1837          if (Start_OStime <= 6) {
1838              if (Ta1[Start_OStime] < Ta2[Start_OStime] && Ta1[Start_OStime] < Ta3[Start_OStime]) {
1839                  if (now_Hiprio != 1) { ptcb = ptcb->OSTCBNext; now_Hiprio = 1; }
1840              }
1841              if (Ta2[Start_OStime] < Ta1[Start_OStime] && Ta2[Start_OStime] < Ta3[Start_OStime]) {
1842                  if (now_Hiprio != 2) { ptcb = ptcb->OSTCBNext; now_Hiprio = 2; }
1843              }
1844              if (Ta3[Start_OStime] < Ta1[Start_OStime] && Ta3[Start_OStime] < Ta2[Start_OStime]) {
1845                  if (now_Hiprio != 3) { ptcb = ptcb->OSTCBNext; now_Hiprio = 3; }
1846              }

```



[PART III] Performance Analysis

1.Compare the scheduling behaviors between NPCS and CPP with the results of PART I and PART II. (5%)

NPCS :

If a job is using a resource, it won't be preempted by any other jobs

- Even if there are no resource conflicts
- Effectively the job runs at the highest priority

CPP :

If R is in use, T is blocked.

If R is free, R is allocated to T. T's execution priority is raised to the priority ceiling of R if that is higher. At any given time, T's execution priority equals the highest priority ceiling of all its held resources.

2.Explain how NPCS and CPP avoid the deadlock problem. (5%)

NPCS :

Deadlocks never occur because any job holding resources can not be preempted

CPP :

T's priority is assigned the next-highest priority ceiling of another resource when the resource with the highest priority ceiling is released. The task returns to its assigned priority after it has released all resources.