Operating System Concepts

Che-Wei Chang

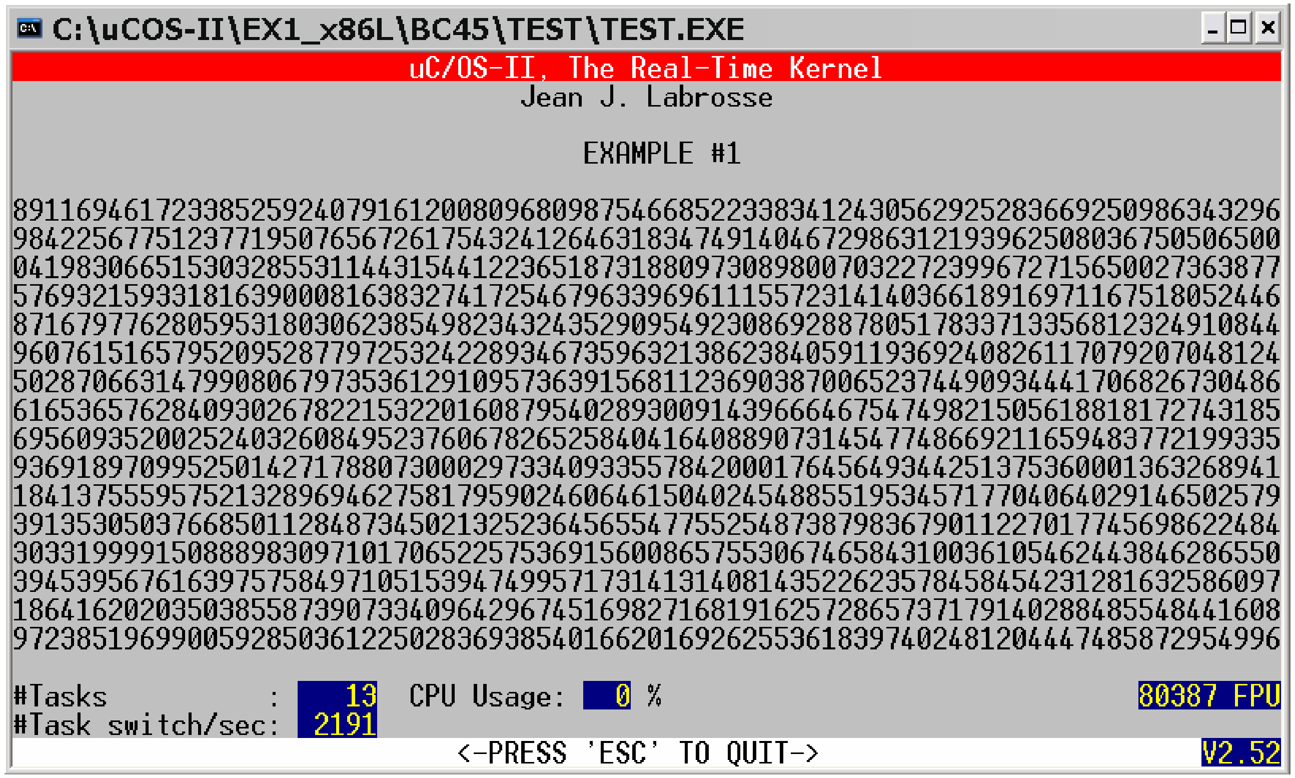
chewei@mail.cgu.edu.tw

Department of Computer Science and Information Engineering, Chang Gung University

Homework 5 (Final Project)-Exercise on μC/OS-II

Example 1 on the Textbook

An Example on μC/OS-II: Multitasking

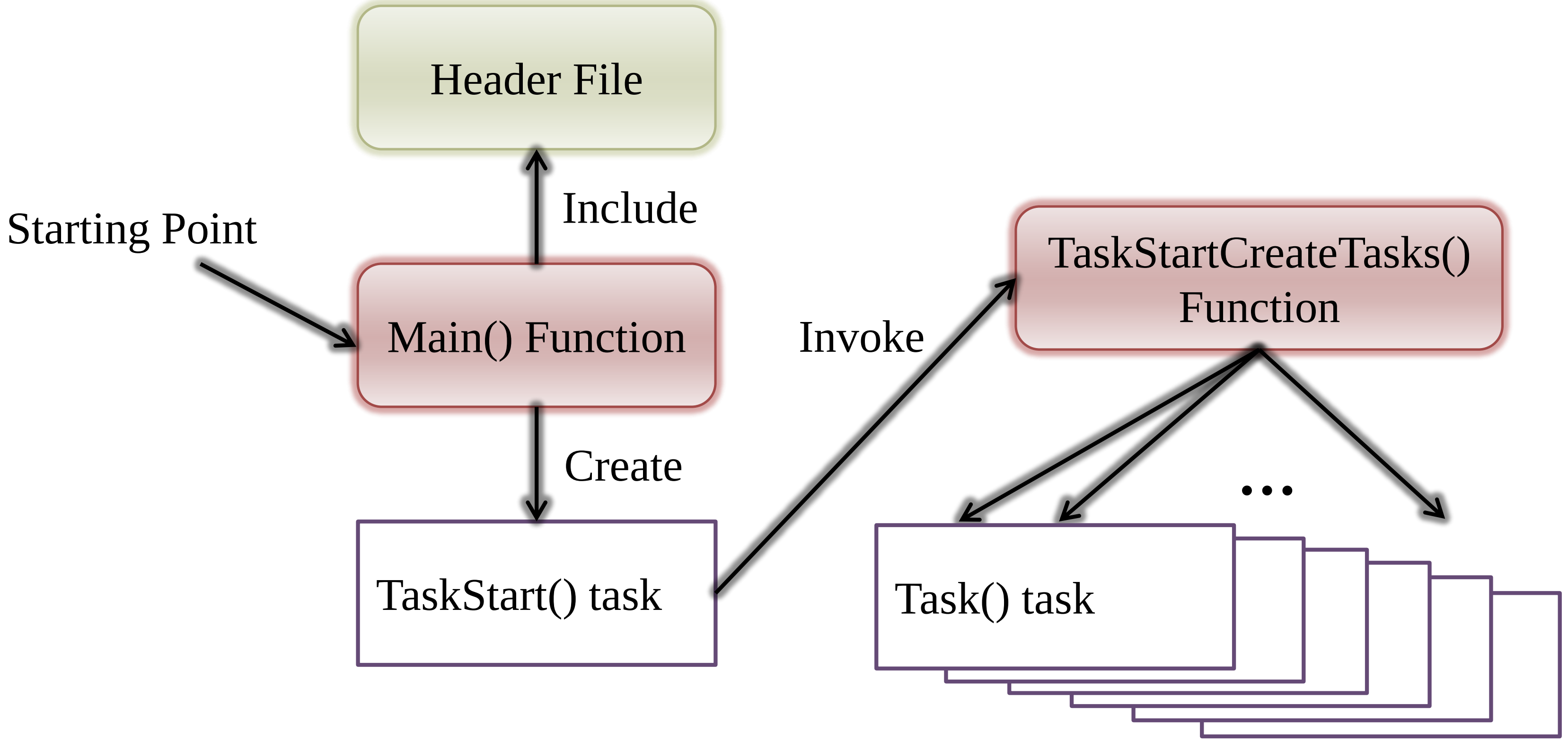


C:\uCOS-II\EX1_x86L\BC45\TEST\TEST.EXE

```
                        uC/OS-II, The Real-Time Kernel
                               Jean J. Labrosse

                                  EXAMPLE #1

89116946172338525924079161200809680987546685223383412430562925283669250986343296
98422567751237719507656726175432412646318347491404672986312193962508036750506500
04198306651530328553114431544122365187318809730898007032272399672715650027363877
57693215933181639000816383274172546796339696111557231414036618916971167518052446
87167977628059531803062385498234324352909549230869288780517833713356812324910844
96076151657952095287797253242289346735963213862384059119369240826117079207048124
50287066314799080679735361291095736391568112369038700652374490934441706826730486
61653657628409302678221532201608795402893009143966646754749821505618818172743185
69560935200252403260849523760678265258404164088907314547748669211659483772199335
93691897099525014271788073000297334093355784200017645649344251375360001363268941
18413755595752132896946275817959024606461504024548855195345717704064029146502579
39135305037668501128487345021325236456555477552548738798367901122701774569862248
30331999915088898309710170652257536915600865755306746584310036105462443846286550
39453956761639757584971051539474995717314131408143522623578458454231281632586097
18641620203503855873907334096429674516982716819162572865737179140288485548441608
97238519699005928503612250283693854016620169262553618397402481204447485872954996

#Tasks          :    13   CPU Usage:    0 %                                 80387 FPU
#Task switch/sec:  2191
                         <-PRESS 'ESC' TO QUIT->                              V2.52
```

- Three system tasks
- Ten application tasks randomly prints its number

Multitasking: Workflow



Multitasking: TEST.C

(\SOFTWARE\uCOS-II\EX1_x86L\BC45\SOURCE\TEST.C)

```
#include "includes.h"
/*
*****************************************************************
CONSTANTS
*****************************************************************
*/
#define TASK_STK_SIZE 512
#define N_TASKS 10
/*
*****************************************************************
VARIABLES
*****************************************************************
*/
OS_STK TaskStk[N_TASKS][TASK_STK_SIZE];
OS_STK TaskStartStk[TASK_STK_SIZE];
char TaskData[N_TASKS];
OS_EVENT *RandomSem;
```

Multitasking: Main()

```
void main (void)
{
        PC_DispClrScr(DISP_FGND_WHITE + ISP_BGND_BLACK);
        OSInit();
        PC_DOSSaveReturn();
        PC_VectSet(uCOS, OSCtxSw);
        RandomSem = OSSemCreate(1);
        OSTaskCreate( TaskStart,
                      (void *)0,
                      (void *)&TaskStartStk[TASK_STK_SIZE-1],
                      0);
        OSStart();
}
```

Entry point of the task (a pointer to a function)

User-specified data

Top of stack

Priority (0=hightest)

Multitasking: TaskStart()

```
void TaskStart (void *pdata)
{
        /*skip the details of setting*/
        OSStatInit();
        TaskStartCreateTasks();
        for (;;)
        {
                if (PC_GetKey(&key) == TRUE)
                {
                        if (key == 0x1B) { PC_DOSReturn(); }
                }
                OSTimeDlyHMSM(0, 0, 1, 0);
        }
}
```

Call the function to create the other tasks

See if the ESCAPE key has been pressed

Wait one second

Multitasking: TaskStartCreateTasks()

```
static void TaskStartCreateTasks (void)
{
        INT8U i;
        for (i = 0; i < N_TASKS; i++)
        {
                TaskData[i] = '0' + i;
                OSTaskCreate(
                        Task,
                        (void *)&TaskData[i],
                        &TaskStk[i][TASK_STK_SIZE - 1],
                        i + 1 );
        }
}
```

Entry point of the task (a pointer to function)

Argument: character to print

Top of stack

Priority

Multitasking: Task()

```
void Task (void *pdata)
{
        INT8U x;
        INT8U y;
        INT8U err;
        for (;;)
        {
                 OSSemPend(RandomSem, 0, &err);
                /* Acquire semaphore to perform random numbers */
                x = random(80);
                /* Find X position where task number will appear */
                y = random(16);
                /* Find Y position where task number will appear */
                OSSemPost(RandomSem);
                /* Release semaphore */
                PC_DispChar(x, y + 5, *(char *)pdata, DISP_FGND_BLACK +DISP_BGND_LIGHT_GRAY);
                /* Display the task number on the screen */
                OSTimeDly(1);
                /* Delay 1 clock tick */
        }
}
```

Randomly pick up the position to print its data

Print & delay

OSinit()

(\SOFTWARE\uCOS-II\SOURCE\OS_CORE.C)

- Initialize the internal structures of µC/OS-II and MUST be called before any services
- Internal structures of µC/OS-2
  - Task ready list
  - Priority table
  - Task control blocks (TCB)
  - Free pool
- Create housekeeping tasks
  - The idle task
  - The statistics task

PC_DOSSaveReturn()
(\SOFTWARE\BLOCKS\PC\BC45\PC.C)

- Save the current status of DOS for the future restoration
  - Interrupt vectors and the RTC tick rate
- Set a global returning point by calling setjump()
  - μC/OS-II can come back here when it terminates.
  - PC_DOSReturn()

PC_VectSet(uCOS,OSCtxSw)
(\SOFTWARE\BLOCKS\PC\BC45\PC.C)

- Install the context switch handler
- Interrupt 0x08 (timer) under 80x86 family
  - Invoked by INT instruction

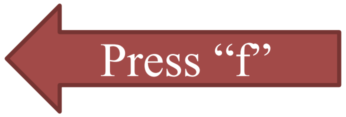
OSStart()
(SOFTWARE\uCOS-II\EX1_x86L\BC45\SOURCE\CORE.C)

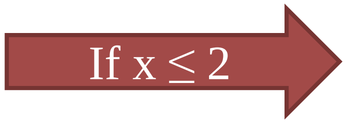
- Start multitasking of µC/OS-II
- It never returns to main()
- µC/OS-II is terminated if PC_DOSReturn() is called

Project Requirements

A Two-Mode Control System

- Normal Mode
  - Show your student ID on the screen
  - Keep changing something on the screen to show the system is active
  - For each second, randomly generate an integer x, 1≤x ≤10, and show it on the screen





- Emergency Mode
  - Count down for 5 seconds
  - Show the remaining time on the screen
  - If no pressing "f" in 5 seconds:
    - Show "System Failure"
    - Delay for 3 seconds
    - Then terminate μC/OS-II

Bonus

- Bonus 1 (10%): Implement the normal mode and emergency mode in different tasks
- Bonus 2 (0%~10%): Implement another mode doing something else

Report

1. The steps for your implementation
2. The problem you met, and how you solved it
3. The bonus you have done
4. **The reference of this homework**

- The report is limited within 4 pages in PDF
  - One bonus, one extra page
- Each bonus you have done, one more page for the report

Grading

- Implementation
  - Implement the two modes 30%
  - The timing behavior is correct 30%
- Report
  - 20%
- Bonus
  - Bonus 1 10%
  - Bonus 2 10%
- Demo Q&A
  - 20%

Submission

- Homework 5 deadline: at 23:00 on 2025-12-16

➔**NO DELAY!**

- Upload to e-learning system
- The title of the report: OSHomework5StudentID
- **Point deduction for wrong format: 10%**

➔DEMO will be arranged!