



Operating System Concepts

Che-Wei Chang

chewei@mail.cgu.edu.tw

Department of Computer Science and Information
Engineering, Chang Gung University



Homework 4– An Real-Time OS: μC/OS-II Quick Overview

Introduction of μC/OS-II (1 / 2)

- ▶ The name is from micro-controller operating system, version 2
- ▶ μC/OS-II is certified in an avionics product by FAA in July 2000 and is also used in the Mars Curiosity Rover
- ▶ It is a very small real-time kernel
 - Memory footprint is about 20KB for a fully functional kernel
 - Source code is about 5,500 lines, mostly in ANSI C
 - Its source is open but not free for commercial usages
- ▶ Preemptible priority-driven real-time scheduling
 - 64 priority levels (max 64 tasks)
 - 8 reserved for μC/OS-II
 - Each task is an infinite loop



Introduction of μC/OS-II (2/2)

- ▶ Deterministic execution times for most μC/OS-II functions and services
- ▶ Nested interrupts could go up to 256 levels
- ▶ Supports of various 8-bit to 64-bit platforms: x86, ARM, MIPS, 8051, etc.
- ▶ Easy for development: Borland C++ compiler and DOS (optional)
- ▶ However, uC/OS-II still lacks of the following features:
 - Resource synchronization protocol
 - Soft-real-time support

The µC/OS-II File Structure

Application Code (Your Code!)

Processor Independent Implementations

- Scheduling policy
- Event flags
- Semaphores
- Mailboxes
- Event queues
- Task management
- Time management
- Memory management

Application Specific Configurations

- OS_CFG.H
- Max # of tasks
- Max Queue length
- ...

uC/OS-II Port for Processor Specific Codes

Software
Hardware

CPU

Timer

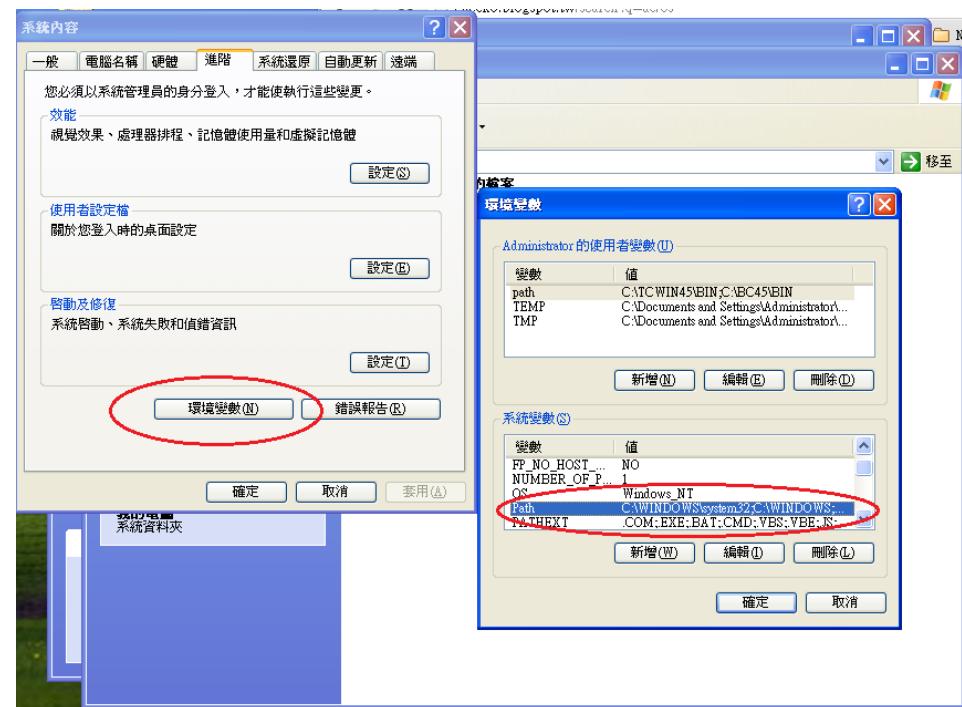
Requirements of μC/OS-II Emulator

- ▶ Operating System
 - Windows XP 32bits
 - Use virtual machine to install the OS
 - Install “Guest Additions” for Virtualbox
- ▶ Tools
 - Borland C++ compiler (V4.5)
 - BC45 is the compiler
 - Turbo Assembler
 - The assembler is in tasm
 - The source code and the emulation environment of μC/OS-II
 - SOFTWARE is the package
- ▶ Full Package
 - Download it from the course website with password: csie2020
 - https://www.csie.cgu.edu.tw/~chewei/files/ucOSII_ProjectPackage.zip
 - <https://www.csie.cgu.edu.tw/~chewei/files/Files.zip>



Borland C++ Compiler

- ▶ Download Borland C++ and install it on your windows XP environment
 - Double click the “INSTALL.EXE”
- ▶ Add “;C:\BC45\BIN” to your system Path



Turbo Assembler

- ▶ Download Turbo assembler and unzip the file
- ▶ Copy “\tasm\BIN\TASM.EXE” to your “C:\BC45\BIN”
 - Include the missing assembler which is going to be used during we compile the source code of µC/OS-II

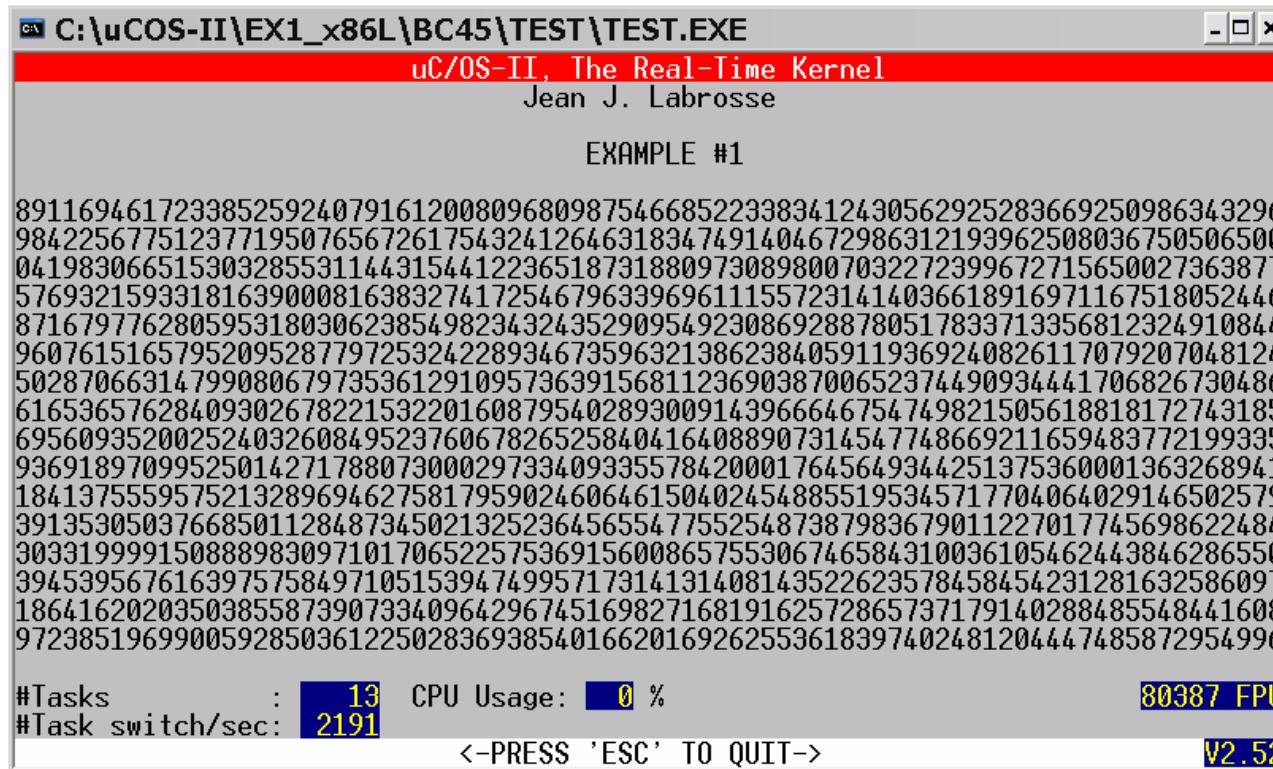
Compile µC/OS-II Example Code

- ▶ Download the source code and emulator µC/OS-II
 - It is recommended to put the source code package “SOFTWARE” directly in C:\
- ▶ Test the first example
 - Execute C:\SOFTWARE\uCOS-II\EX1_x86L\BC45\TEST\TEST.EXE
 - Press ECS to leave
- ▶ Rename or remove the executable file
 - Rename TEST.EXE
- ▶ Compile the µC/OS-II and the source code of the first example
 - Run C:\SOFTWARE\uCOS-II\EX1_x86L\BC45\TEST\MAKETEST.BAT
 - A new “TEST.EXE” will be created if we compile it successfully

Common Mistakes

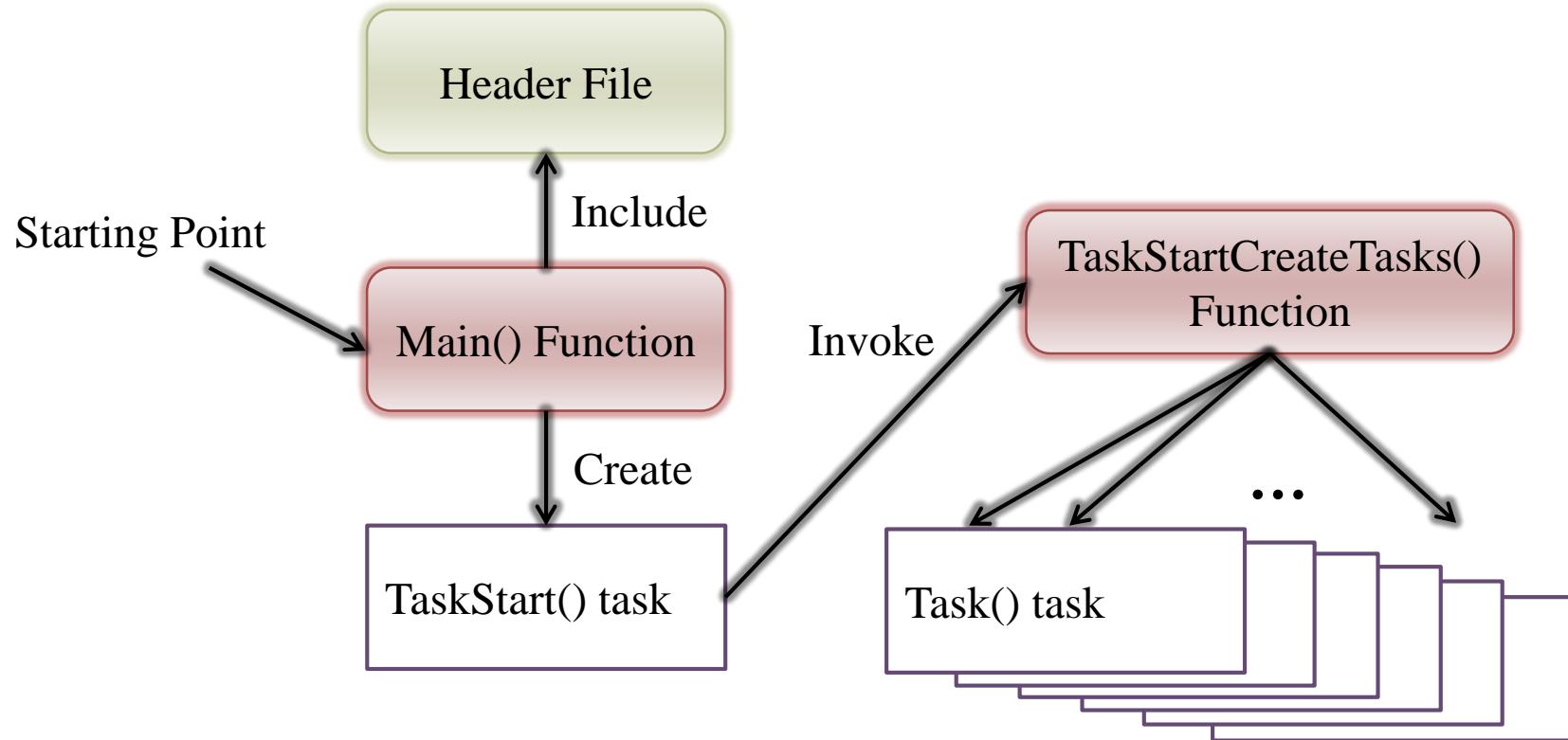
- ▶ Did you directly put the package “SOFTWARE” in C:\ ?
- ▶ Have you copied the correct file “TASM.EXE” to your “C:\BC45\BIN” directory?
- ▶ Did you set the Path correctly?
 - See the picture in Page 7
 - There is no space

An Example on μC/OS-II: Multitasking



- ▶ 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,
```

Top of stack

```
(void *)0,
```

Priority (0=highest)

```
(void *)&TaskStartStk[TASK_STK_SIZE-1],
```

```
    0);
```

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

User-specified data

```
}
```

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 );
```

Top of stack

Priority

Entry point of the task
(a pointer to function)

Argument:
character to print

```
}
```

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 */
    }
}
```

Print & delay

Randomly pick up the position to print its data

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

Report

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

► The report is limited within 4 pages in PDF

Extra Exercise

- ▶ Read the e-book of μC/OS-II
 - Try to read and understand the first chapter
- ▶ Read the source code to understand the application
 - The application source code is in C:\SOFTWARE\uCOS-II\EX1_x86L\BC45\SOURCE
- ▶ Browse the source code of μC/OS-II
 - The source code of μC/OS-II is in C:\SOFTWARE\uCOS-II\SOURCE
- ▶ 準時繳交且實作完成第九頁的內容，提供截圖或相關說明 → 標準分數為80正負10分
- ▶ 有做Extra Exercise，並寫入報告心得且說明精確者最多加20分

Grading

▶ Implementation

- Install the environment for running µC/OS-II 30%
- Compile and run the first example 30%

▶ Report

- 20%

▶ Bonus

- Extra exercise 20%

▶ Demo Q&A

- 20%

Submission

- ▶ Homework 4 deadline: at 23:00 on 2025-12-2
→NO DELAY!
- ▶ Upload to e-learning system
- ▶ The title of the report: OSHomework4StudentID
- ▶ **Point deduction for wrong format: 10%**

→DEMO will be arranged!