



Operating System Practice- Final Project

Che-Wei Chang

chewei@mail.cgu.edu.tw

Department of Computer Science and Information
Engineering, Chang Gung University

Report

- ▶ Only four A4 pages
- ▶ 12 pt words
- ▶ Deadline is 23:59 2020/06/30
- ▶ File name: OSP-Project-GroupID.zip
- ▶ Required Files: The source code files and the report
- ▶ In the report, remember to provide your names, student IDs, and group ID.
- ▶ Send it to TA's email: nasa91011@gmail.com
- ▶ Email title: OSP Project GroupID

The Requirements of Final Presentation

- ▶ Presentation is only for **6 minutes**
 - Quickly go through the implementation
 - Talk more about the problems you solved
 - Highlight your extra exercise
- ▶ Live demo is required
 - Bring your source code
- ▶ I will ask each of you a question
 - You have **30 seconds** to answer the question

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
- ...

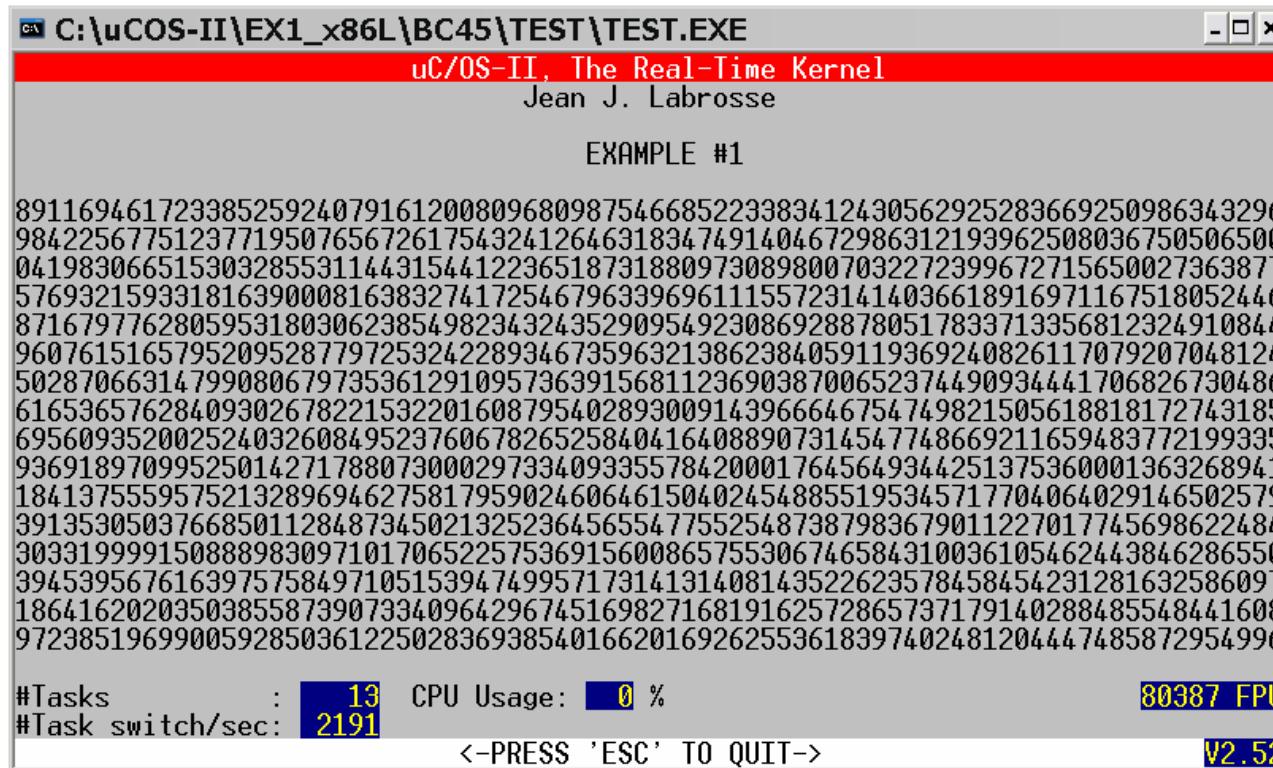
uC/OS-II Port for Processor Specific Codes

Software
Hardware

CPU

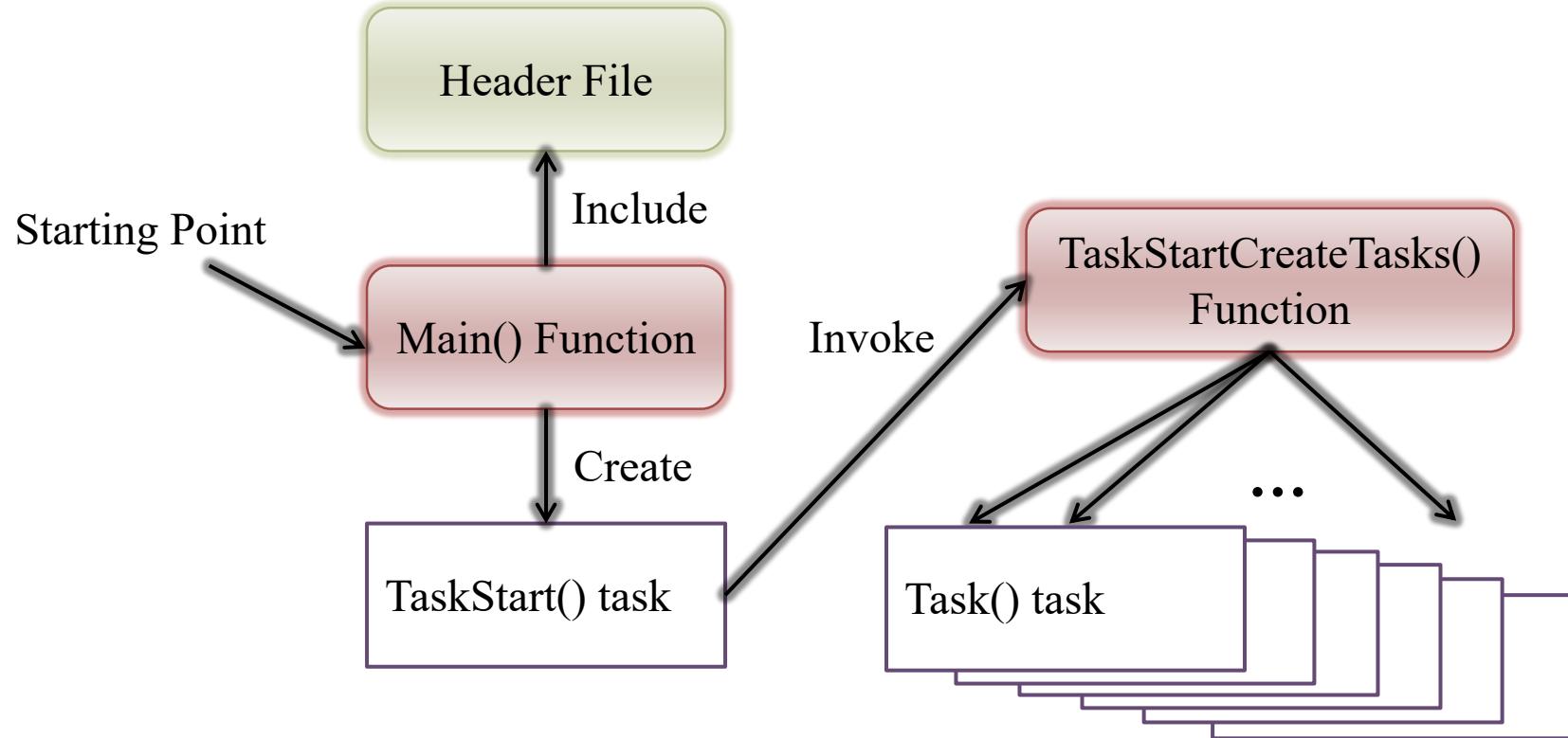
Timer

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,
                  (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=highest)

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

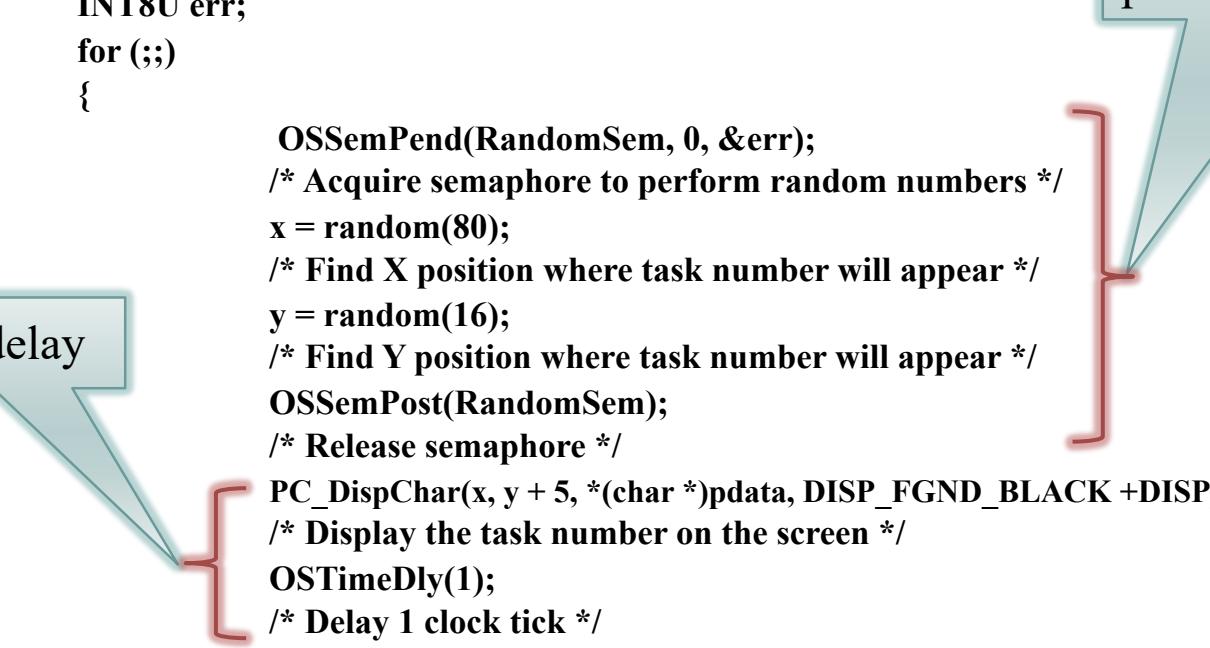
```
}
```

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_DispcChar(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

Requirements

- ▶ Task Scheduling
 - Adopt priority-driven scheduling
 - The scheduler always schedules the highest priority ready task to run
 - Modify the priority of each task
 - Related code in uC/OS II
 - See OS_Sched() for scheduling policy
 - See OSTimeTick() for time management
 - See OSIntExit() for the interrupt management
- ▶ Provide the RM/EDF Scheduler
 - Input: A task set, each task is with its execution time and period
 - Output: The printed result of each task

Input

- ▶ The input format should be as follows
 - Your program should have the capability to create the assigned number of tasks and their corresponding period and execution time.
 - Example: taskset.txt

```
3 //number of task  
1 3 // task 1: (execution time 1, period 1)  
2 9 // task 2: (execution time 2, period 2)  
4 12 // task 3: (execution time 3, period 3)
```
- ▶ The total utilization is no more than **65%**
- ▶ The number of tasks is no more than **7**

Input Example (1 / 2)

4

1 12

1 7

2 19

3 20

Input Example (2/2)

5
1 18
1 17
2 16
1 20
1 6

Output

- ▶ Your program output must show the following information
 - A sequence of the running task over time
 - The time when context switch occurred
- ▶ A report to describe your implementation
 - Relationship of each function
 - Implementation flow chart
 - Implementation details

Hints (1 / 2)

- ▶ You can read three other example in the document and refer to the source code.
- ▶ In order to implement a new scheduler, we might have to modify the `os_tcb` data structure to include some new attributes.
- ▶ The function `OSTaskCreateExt()` is used to create tasks, and we can modify this function to input the execution time and the period to each task.
- ▶ Each task executes an infinite loop and uses `OSTimeGet()` to get the execution time, where `OS_TICKS_PER_SEC` is the number of ticks for a second.
 - Note that a task might be preempted during its execution.
- ▶ Use `OSTimeDly()` when the task finish its execution.

Hints (2/2)

- ▶ Modify the deadline of a task before it call OSTimeDly() (ex: OSTCBCur->deadline=OSTCBCur->deadline+TaskPeriod)
- ▶ When the delay of a task is completed, the function OSTaskResume() is called to put the task back to ready queue and reschedule.
- ▶ Modify the function OS_Sched() to pick the task with the shortest period or the earliest deadline.
- ▶ OSStart() is used to start the execution of tasks.