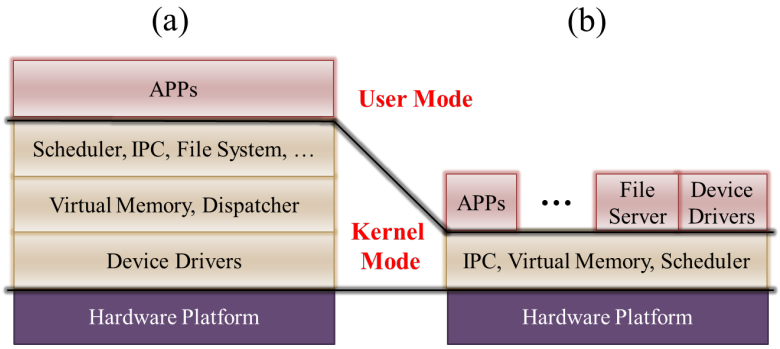
長庚大學106學年度第二學期 作業系統實務 期末測驗 (總分101)

<<請依題號順序作答，跳號作答不予計分>>

系級: 姓名: 學號:

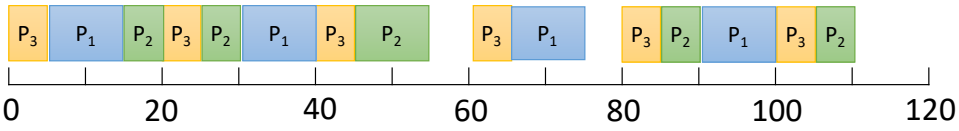
1. (8%) Consider the monolithic kernel and the micro kernel, and refer to the following figure. (a) For the left side, which kernel is it? (b) On the right side of the figure, which kernel is it? (c) List one advantage and one disadvantage of the monolithic kernel.



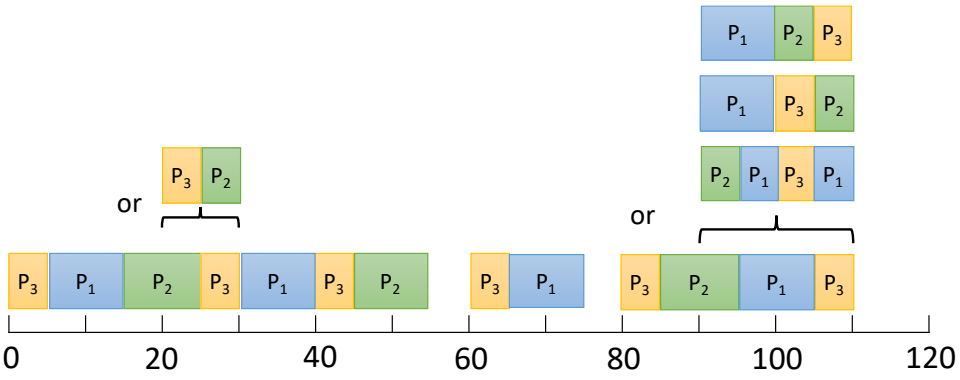
Answer: (a) Monolithic kernel.(2%)
(b) Micro kernel. (2%)
(c) Advantage: All parts of the kernel share the same kernel-level memory for efficient communication. Less IPC calls are required for the kernel functions. In general, the performance is better. (2%)
Disadvantage: The size of a monolithic kernel is usually much larger than that of a micro kernel. Thus, it could be different to maintain, trace, update, and debug a monolithic kernel. (2%)

2. (12%) here are three periodic tasks P_1, P_2 and P_3. P_1 has its period 30 and execution time 10. P_2 has its period 40 and execution time 10. P_3 has its period 20 and execution time 5. Please draw the results of the (a) RM and (b) EDF scheduling algorithms. Assume that all tasks arrive at time 0. Please draw the schedule result from time 0 to time 120 if all tasks can meet their deadlines. Please draw the result from time 0 until the first deadline missing if any task will miss its deadline.

Answer: (a) RM (6%)



(b) EDF (6%)



3. (12%) Consider 4 tasks, t_1, t_2, t_3, and t_4 which have priorities x_1, x_2, x_3, and x_4, respectively, and assume $x_1>x_2>x_3>x_4$ (x_1 is the highest priority). After we profiled the programs of the 4 tasks, we have

the following information:

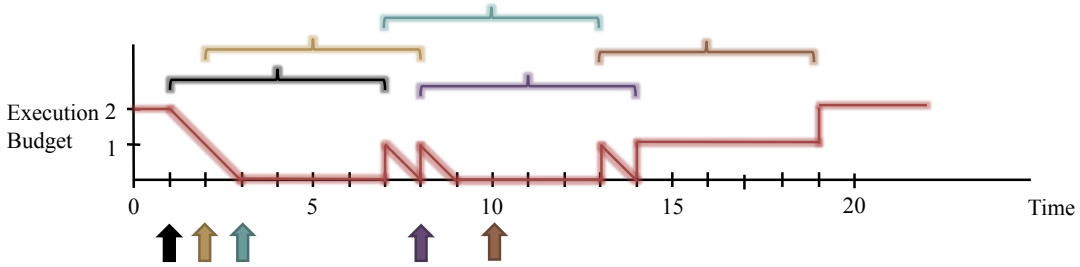
- Task t_1 will lock semaphore S_1 for 20 ms.
- Task t_2 will lock semaphore S_1 for 23 ms and lock semaphore S_2 for 50 ms.
- Task t_3 will lock semaphore S_1 for 7ms and lock semaphore S_3 for 5ms.
- Task t_4 will lock semaphore S_2 for 10ms and lock semaphore S_4 for 13ms.

(a) Please derive the priority ceiling of each semaphore. If the priority ceiling protocol is used to manage the semaphore locking, (b) please derive the worst-case blocking time of each task. You have to provide the reason to support each of your answers.

Answer: (a) S_1: x_1, S_2: x_2, S_3: x_3, S_4: x_4.. You have to provide the season to support your answers.
(b) t_1: 23ms, t_2: 10ms, t_3: 10ms, t_1: 0ms.You have to provide the season to support your answers.

4. (12%) A sporadic server has a replenishment period 6 and an execution budget 2. Let the sporadic server have the budget 2 at time 0. Assume that events arrive at 1, 2, 3, 8, 10, and each event consumes the execution time 1. Please draw a diagram to show the changing of the execution budget at different time points.

Answer:



5. (8%) Please define "Priority Inversion". A good idea is to use an example to show the concept of Priority Inversion.

Answer: When a high-priority task is blocked by a low-priority task, and the low priority task is further preempted by some medium-priority tasks, there is priority inversion. The high-priority task is indirectly preempted by the medium-priority tasks and suffers long waiting (and blocking) time.

6. (8%) Worst-Case Execution Time (WCET) analysis is very important and essential for real-time system designs. In practice, we usually use an upper bound of the WCET instead of a real WCET. Why do we need to use the upper bound?

Answer: It is (theoretically) impossible to derive the WCETs for all programs. Thus, using the upper bound of the WCET of a program can be safe to make sure that the program can be completed before its deadline.

7. (8%) The code of µC/OS-II for creating new tasks in Example 1 is as follows. Please fill (a) and (b)

```
static void TaskStartCreateTasks (void)
{
    INT8U i;
    for (i = 0; i < N_TASKS; i++)
    {
        TaskData[i] = '0' + i;
        OSTaskCreate(
            Task,                                   (a)
            (void *)&TaskData[i],                   User-specified data:
            &TaskStk[i][TASK_STK_SIZE - 1],         A character to print
                                                    Top of stack
            i + 1 );                                (b)
    }
}
```

Answer: (a) The entry point of the task (a function pointer)
(b) The priority of the task

8. (8%) In our lab exercises, we need a “cross compiler” to compile the Linux kernel (in Lab 1) and kernel modules (in Lab 2). Why do we need to use the cross compiler instead of the built-in normal GCC compiler?

Answer: We have to compile the source code on an Intel X86 computer but run the binary (program) on an ARM evaluation board. The ordinary gcc on an X86 computer will generate binaries for X86. Thus, we need a cross compiler to generate binaries for ARM architecture.

9. (9%) Please explain the purposes of using (a) TFTP, (b) NFS, and (c) Minicom in our lab exercises.

Answer: (a) TFTP is used to download the kernel image from our PCs to the evaluation boards.
(b) NFS is used to share the root filesystem (files) on our PCs to the evaluation boards.
(c) Minicom is used to pass the commands to the evaluation boards and to forward the results on the evaluation boards to our PCs.

10. (16%) Happy Coding Time! On the left, you have a sample kernel module. On the right, you have anything else you need in this question. Please modify the code of the kernel module to meet the following requirements: (1) Make LED 1 “BLINK” at the beginning of the insertion. (2) After 5 seconds of the insertion, make LED 1 “ON”. (3) Make LED 1 “BLINK” at the beginning of the module deletion. (4) After 4 seconds of the deletion, make LED 1 “OFF”.

```
#include <linux/init.h>
#include <linux/module.h>
MODULE_LICENSE("License for you");

static int mymodule_init(void)
{
        printk("Instert My Module!\n");
        return 0;
}

static void mymodule_exit(void)
{
        printk("My Module is Unloaded!\n");
}

module_init(mymodule_init);
module_exit(mymodule_exit);
```

- The header file and function of LED
  - #include <asm-arm/arch-omap/tps65010.h>
  - tps65010_set_led(which_LED , which_operation);
- The LEDs on the board
  - LED1 and LED2
- The operations of a LED
  - OFF, ON, or BLINK
- The header file and macro Timer
  - #include <linux/timer.h>
  - 『jiffies』 is the counter for timer interrupts
  - 『HZ』 is the number of timer interrupts per second

Answer:

```
#include <linux/init.h>
#include <linux/module.h>
#include <asm-arm/arch-omap/tps65010.h>
#include <linux/timer.h>
MODULE_LICENSE("License for you");
int a;
static int mymodule_init(void)
{
        tps65010_set_led(LED1 , BLINK);
        a = jiffies;
        while(a + 5*HZ > jiffies) ;
        tps65010_set_led(LED1 , ON);
        return 0;
}
static void mymodule_exit(void)
{
        tps65010_set_led(LED1 , BLINK);
        a = jiffies;
        while(a + 4*HZ > jiffies) ;
        tps65010_set_led(LED1 , OFF);
}
module_init(mymodule_init);
module_exit(mymodule_exit);
```