長庚大學108學年度第二學期 作業系統實務 期末測驗（總分106）

<<請依題號順序作答，跳號作答不予計分>>

系級:　　　　　　　姓名:　　　　　　　學號:

1. (8%) What is the “proc” filesystem of Linux?

Answer: The proc filesystem is a special filesystem in Unix-like operating systems that presents information about processes and other system information in a hierarchical file-like structure, providing a more convenient and standardized method for dynamically accessing process data held in the kernel than traditional tracing methods or direct access to kernel memory. It can be used to collect some hardware and system information via some user file I/O requests.

2. (8%) A Linux kernel module can run as a part of the Linux kernel, and for some functions, we prefer to implement them in Linux kernel modules instead of implementing them directly into the Linux kernel. Please explain the behavior and characteristics of “Linux Kernel Module.”

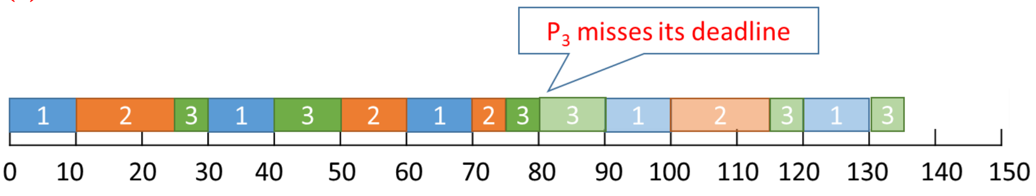
Answer: Since the Linux kernel is a monolithic kernel, it needs some mechanism to dynamically insert, update, and remove some function to enhance its flexibility. A Linux kernel module can be inserted to and removed from a Linux kernel without rebooting the system, and after the module insertion, the kernel module can act as a part of the Linux kernel. Thus, it is a very convenient and common approach for including new functions into the Linux kernel without shutting down the system.

3. (16%) Please briefly explain (a) the **Rate Monotonic (RM)** scheduling algorithm and (b) the **Earliest Deadline First (EDF)** scheduling algorithm for real-time task scheduling. For 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 50 and execution time 15, and P_3 has its period 75 and execution time 20. Assume P_1 ,P_2 and P_3 are ready at time 0. Please draw the scheduling results from time 0 to 150 for (c) the RM scheduling and (d) the EDF scheduling.

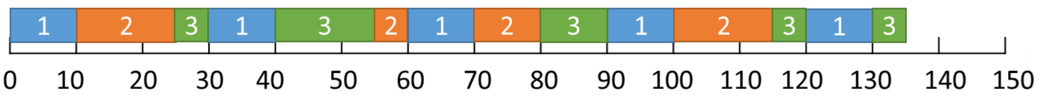
Answer: (a) A task with a shorter period will be assigned a higher priority.

(b) A task with the earliest deadline will be assigned the highest priority.

(c)



(d)



4. (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 40 ms.
   - Task t_2 will lock semaphore S_1 for 20 ms and lock semaphore S_2 for 50 ms.
   - Task t_3 will lock semaphore S_2 for 10ms and lock semaphore S_3 for 60ms.
   - Task t_4 will lock semaphore S_3 for 30ms and lock semaphore S_4 for 70ms.

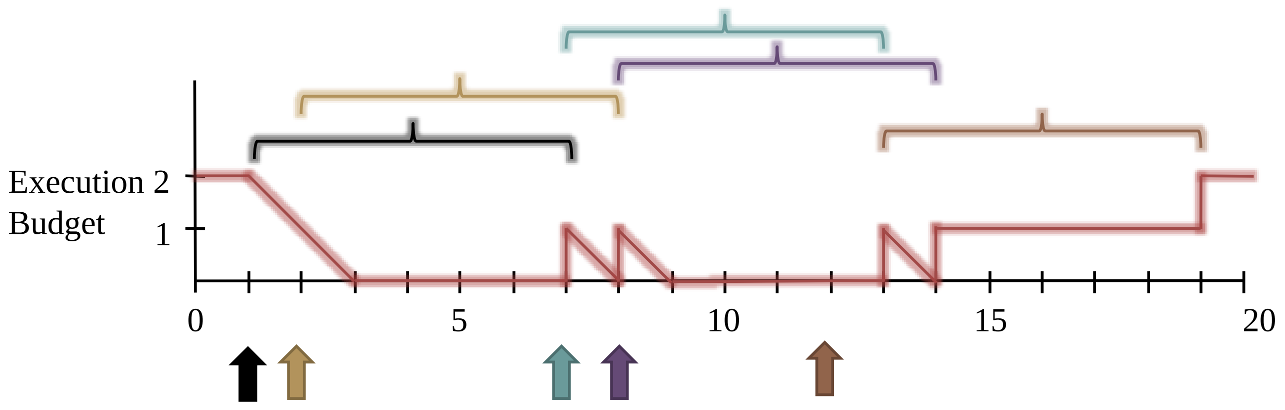
(a) Please derive the priority ceiling of each semaphore. If the priority ceiling protocol is used to manage the semaphore locking, (b) please derive the worst-case blocking time of each task. You have to provide the reason to support each of your answers.

Answer: (a) Priority ceilings: S_1: x_1, S_2: x_2, S_3: x_3

(b) Worst-case blocking times: t_1: 20 ms, t_2: 10 ms, t_3: 30 ms, t_4: 0 ms.

5. (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, 7, 8, 12, and each event consumes the execution time 1. Please draw a diagram from time 0 to time 20 to show the changing of the execution budget at different time points.

Answer:



6. (10%) Why do we need to have the WCET (worst case execution time) upper bound of each real-time task before we do real-time scheduling?

Answer: To do real-time scheduling, we would like to know whether a real-time task is always completed before the deadline. Since the execution time of the task might not be a constant, we have to know the longest possible execution time, i.e., WCET. Since it might be impossible to derive the WCET of the task, we then use a WCET upper bound to represent the WCET.

7. (10%) For WCET analysis, an important job is to do cache analysis. Assume that only basic block A and basic block B might call basic block C, and the LRU (least recently used) policy is used for the cache management. We know that the cache state after running basic block A is {{a},{b},{c},{d}}, where {a} is the most recently used data. The cache state after running basic block B is {{ },{b},{a, d},{e}}. Please provide the cache state before we run basic block C.

Answer: {{},{b},{a},{d}}

8. (10%) When the OS **μC/OS-II** is running, (a) can we directly install a new application on it? (b) If your answer is yes, please explain the procedure for installing a new application. If your answer is no, please describe how can we have a new application on **μC/OS-II**.

Answer: (a) No (4%)

(b) We have to collect the source files of μC/OS-II and all applications (including the new application) and compile them to have the new image. The system should be turned off, and the image is then install on the system. After that, we can boot the system with the new application running. (6%)

9. (10%) In Lab 1, (a) why do we have to set up the TFTP connection between the PC and the evaluation board? (b) Why do we have to build the NFS server on the PC for the evaluation board?

Answer: (a) The TFTP is used to download the Linux kernel from our PC to the evaluation board.
(b) The NFS is used to provide the root file system for the evaluation board such that we can direct put the to-be-executed program on the root file system and direct execute the program on the evaluation board.

10. (10%) In Lab 2, to control the LED, we have to include the header file:
#include <asm-arm/arch-omap/tps65010.h>
And the following function can be used to control the LED:
tps65010_set_led(which_LED , what_operation);
There are two LED devices defined in the header file: *LED1* and *LED2*.
There are three operations for a LED device: *OFF*, *ON*, and *BLINK*.
Please modify the following source code of our kernel module to

- make LED1 and LED2 ON when the module is loaded, and
- make LED1 and LED2 OFF when the module is unloaded

```
#include <linux/init.h>
#include <linux/module.h>
MODULE_LICENSE("License for you");
static int OSP_Pass(void) { return 0; }
static void Summer_Vacation(void) { }
module_init(OSP_Pass);
module_exit(Summer_Vacation);
```

Answer:

```
#include <linux/init.h>
#include <linux/module.h>
#include <asm-arm/arch-omap/tps65010.h>
MODULE_LICENSE("License for you");
static int OSP_Pass(void)
{
        tps65010_set_led(LED1, BLINK);
        tps65010_set_led(LED2, BLINK);
        return 0;
}
static void Summer_Vacation(void)
{
        tps65010_set_led(LED1, OFF);
        tps65010_set_led(LED2, ON);
}
module_init(OSP_Pass);
module_exit(Summer_Vacation);
```