EURECOM
Sophia Antipolis

Exam
Operating Systems - OS

Ludovic Apvrille
ludovic.apvrille@telecom-paristech.fr

February, 10th, 2012

Authorized documents: Nothing! The exam takes into account that you don't have the right to have any document with you.
A grade is provided for every question (beware: do organize your time). 1 additional point is given as a general appreciation, including written skills and readability.

1 Understanding of the course (10 points, ∼60 minutes)

1. Consider a user-level process executing on a CPU. Give several reasons for this process to be blocked, and so to be preempted from the CPU by the OS? What are the conditions for that process to be granted again the CPU? [2.5 points]

2. Recall what is the Shortest Job First scheduling policy, and explain its main advantages and drawbacks. [2.5 points]

3. What is the purpose of a page replacement algorithm i.e., in which conditions is it used by an OS? Do you remember or can you imagine a few algorithms? If so, explain them briefly. [3 points]

4. Provide the reasons why it is not a good idea to enter a critical section CS2 when already being in a critical section CS1. Give concrete examples to illustrate your response. [2 points]

2 Programming kernels (10 points, ∼60 minutes)

The Linux code provided at the end of this exercise is taken from an IBM webpage whose purpose is to illustrate the use of Linked Lists at kernel level. You goal is to answer the following questions regarding that code.

1. Cite different reasons why linked lists are commonly used in kernels. In particular, could you cite modules of kernel that make use of them? [1.5 points]

2. Apart from the list manipulation, give all the elements of the code that are specific to a (Linux) kernel code, and explain them. Also, explain **WHY** they are specific to kernel code. [1.5 points]

3. Basically explain how the code works. To do so, give the comments you would put in that code to help readers/users of that code to understand it. [1.5 points]

4. To facilitate the use of linked lists for kernel programmers, Linux defines a set of macros. For the following ones, your job is to try to guess their role. Justify your answer, otherwise no point is given. [2.5 points]

   a. LIST_HEAD

   b. list_for_each

   c. list_entry

   d. list_for_each_entry

   e. list_for_each_safe

5. Now, just assume that you have to program the same application in user mode. Explain the general approach, and give the main parts of the C code of that application. [3 points]

(The code is on next page)

```c
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/list.h>

MODULE_LICENSE("GPL");

struct my_data_struct {
  int value;
  struct list_head full_list;
  struct list_head odd_list;
};

LIST_HEAD( my_full_list );
LIST_HEAD( my_odd_list );

int init_module( void ) {
  int count;
  struct my_data_struct *obj;

  for (count = 1 ; count < 11 ; count++) {

    obj = (struct my_data_struct *)
            kmalloc( sizeof(struct my_data_struct), GFP_KERNEL );

    obj->value = count;

    list_add( &obj->full_list, &my_full_list );

    if (obj->value & 0x1) {
      list_add( &obj->odd_list, &my_odd_list );
    }

  }
  return 0;
}

void cleanup_module( void ) {
  struct list_head *pos, *q;
  struct my_data_struct *my_obj;

  printk("Emit full list\n");
  list_for_each( pos, &my_full_list ) {
    my_obj = list_entry( pos, struct my_data_struct, full_list );
    printk( "%d\n", my_obj->value );
  }

  printk("Emit odd list\n");
  list_for_each_entry( my_obj, &my_odd_list, odd_list ) {
    printk( "%d\n", my_obj->value );
  }

  printk("Cleaning up\n");
  list_for_each_safe( pos, q, &my_full_list ) {
    struct my_data_struct *tmp;
    tmp = list_entry( pos, struct my_data_struct, full_list );
    list_del( pos );
    kfree( tmp );
  }

  return;
}
```