Pintos Task 0

211 Operating Systems
Department of Computing
Imperial College

This task is logically divided into two parts:

- Questions 1-8 test your understanding of Pintos basic concepts. To answer these questions you are invited to carefully read the manual. For some questions, examining the Pintos code can be useful to provide exhaustive answers, which might be awarded with extra marks accordingly. The maximum possible mark in Questions 1-8 is 20.
- Question 9 tests your comprehension and ability of writing Pintos code. To successfully accomplish this task, you first have to download, compile, install and run Pintos, and then develop a simple functionality. The maximum possible mark in Question 9 is 20.

Question 1 - (1 mark)

Which Git command should you run to retrieve a copy of your group's shared Pintos repository in your local directory?

Question 2 - (2 marks)

Why is using the `strcpy()` function to copy strings usually a bad idea?
(*Hint: identify the problem, give details and discuss possible solutions.*)

Question 3 - (6 marks)

Explain how thread scheduling in Pintos currently works in less than 300 words. Include the chain of execution of function calls.
(*Hint: we expect you to at least mention which functions participate in a context switch, how they interact, how and when the thread state is modified, the role of interrupts.)*

Question 4 - (2 marks)

Explain the property of reproducibility and how the lack of reproducibility will affect debugging.

Question 5 - (2 marks)

How would you print an unsigned 64 bit `int`? (Consider that you are working with C99). Don't forget to state any inclusions needed by your code.

Question 6 - (3 marks)

Describe the data structures and functions that locks and semaphores in Pintos have in common. What extra property do locks have that semaphores do not?

Question 7 - (3 marks)

In Pintos, a thread is characterized by a struct and an execution stack. What are the limitations on the size of these data structures? Explain how this relates to stack overflow and how Pintos identifies it.

Question 8 - (1 mark)

If test `src/tests/threads/alarm-multiple` fails, where would you find its output and result logs? Provide both paths and file names. (*Hint: you might want to run this test and find out.*)

Question 9 - (20 marks) - The Alarm Clock

In this question, you are requested to implement a simple functionality in Pintos and to answer the questions below.

Coding the Alarm Clock in Pintos

Reimplement `timer_sleep()`, defined in '`devices/timer.c`'. Although a working implementation of `timer_sleep()` is provided, it "busy waits," that is, it spins in a loop checking the current time and calling `thread_yield()` until enough time has gone by. Reimplement it to avoid busy waiting (**10 marks**). Further instructions and hints can be found in the Pintos manual.

You also need to provide a design document which answers the following questions:

Data Structures

A1: (**2 marks**) Copy here the declaration of each new or changed '`struct`' or '`struct`' member, global or static variable, '`typedef`', or enumeration. Identify the purpose of each in 25 words or less.

Algorithms

A2: (**2 marks**) Briefly describe what happens in a call to `timer_sleep()`, including the actions performed by the timer interrupt handler on each timer tick.
A3: (**2 marks**) What steps are taken to minimize the amount of time spent in the timer interrupt handler?

Synchronization

A4: (**1 mark**) How are race conditions avoided when multiple threads call `timer_sleep()` simultaneously?
A5: (**1 mark**) How are race conditions avoided when a timer interrupt occurs during a call to `timer_sleep()`?

Rationale

A6: (**2 marks**) Why did you choose this design? In what ways is it superior to another design you considered?