University of Glasgow

Wednesday, 8th May 2024
09:30 - 11:30 BST
Duration: 2 hours
Timed exam – fixed start time

DEGREES OF MSc, MSci, MEng, BEng, BSc, MA and MA (Social Sciences)

ADVANCED SYSTEMS PROGRAMMING COMPSCI 5083

Answer all four questions

This examination paper is an open book, online assessment and is worth a total of 80 marks.

1. **(a)** The ubiquitous use of multithreading and multicore systems has forced programming language designers to develop sophisticated memory models that describe in detail when locked and unlocked writes to memory become visible to other threads. Outline the benefits of having such a memory model in supporting legacy and new code. Discuss whether you believe the use of shared memory, following the semantics defined in these memory models, provides a suitable basis for future software development. [10]

   (b) Garbage collection is widely used in applications programming but is generally not used by systems programs. Discuss whether it would be possible, in principle, to implement an operating system, including the device drivers and the network stack, in a garbage collected language. Highlight areas of concern or where special treatment might be needed. [10]

2. **(a)** The Rust programming language uses a number of different pointer types and ownership tracking to achieve memory safety and prevent data races. This provides stronger safety guarantees than many other programming languages, but at the expense of adding complexity to the language and limiting the range of data structures that can be readily expressed. Is the incremental gain in safety worth the complexity? Briefly discuss the trade-offs to justify your answer. [5]

   (b) The C programming language permits data to be read from the network into a `char *` buffer then cast to an appropriate type representing the packet format. State three reasons why this is unsafe. [3]

   (c) The safe subset of the Rust language doesn't have the concept of a null pointer, and instead relies on use of the `Option<T>` to indicate missing data. Explain why the use of Option types is, in principle, safer than the use of null pointers. Briefly discuss whether it is safer in practice. [4]

   (d) The lectures discussed two different ways of representing state machines in Rust. The first uses `enum` types to represent states and events that cause state transitions with a function to represent the event-action mapping. The second uses `struct` types to represent the state machine, with the state transitions implemented as methods of that `struct` that consume variables representing the previous state and return representations of the new state. Discuss when you would prefer to use each approach. [8]

3. **(a)** To avoid the problems inherent in languages that use multiple threads with shared mutable state protected by locks, some languages enable concurrency by isolating tasks and providing explicit message passing features. Explain why this approach does not protect from deadlocks and race conditions. Discuss whether you think deadlocks and race conditions are more or less likely to occur than in systems using message passing compared to those using lock-based synchronisation, justifying your answer. [10]

   (b) The Rust programming language provides support for asynchronous programming using the `async` and `await` syntax. Other languages implement similar features. With appropriate runtime support, this aims to provide efficient support for concurrent task execution. Describe, with justification, the types of task that this form of asynchronous programming is intended to support and those that it does not support effectively. Discuss to what extent you believe this style of programming fits with the Rust approach of using strong typing and safe defaults to prevent run-time errors. [10]

4. (a) The lectures reviewed the concept of type driven development and discussed various ways in which the use of strong type systems is claimed to improve the robustness of systems programs, both by better modelling features of the problem domain, and by providing type-level support to detect common classes of bugs.

Discuss whether you believe the claimed benefits are likely to be realised in practice for systems programs. Your answer should reflect on the extent to which improved modelling of the problem domain or eliminating certain classes of bugs are more likely to improve systems program quality. You should highlight the strengths and weaknesses of languages, such as Rust, that claim to support this style of programming and discuss the cost-benefit trade-offs inherent in the use of strong type systems for systems programs. Give examples to illustrate key points. [20]