University of Glasgow

Monday, 28th April 2025
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 worth a total of 80 marks.

1. **(a)** The lectures and laboratory exercises considered two ways of building state machines in the Rust programming language, one using `enum` types to describe states and transitions, the other using `struct` types to model states with functions implemented on those structures to represent transitions. Describe the advantages and disadvantages of the two approaches and discuss whether you think one approach or the other is superior. [9]

(b) The type system of the Rust programming language tracks data ownership and ties memory management to ownership. It has been suggested that this makes Rust easier for C programmers to learn than for Java programmers. Discuss whether you agree that this might be the case, explaining your reasoning. [5]

(c) Systems programs have generally been written in memory unsafe languages, such as C. It is generally accepted that this is problematic, and that it's desirable to move to memory safe languages where possible. Discuss whether it is possible to write an operating system partially or entirely in a memory safe language. Give examples highlighting areas where doing so would be difficult. [6]

2. **(a)** Studies have shown that most security vulnerabilities in deployed systems are due to a lack of memory safety. The lectures described the classic, stack-smashing, buffer overflow attack as an example of a memory safety vulnerability. Briefly describe how this attack works and suggest ways in which it can be mitigated. Assume that it is not possible to replace the vulnerable program with an implementation in a memory safe language. [5]

(b) Postel's law states, in the context of network protocol implementation, "Be liberal in what you accept, and conservative in what you send". Explain why Postel's law is no longer appropriate in today's networks. Discuss how code that parses network data should be organised, and how the use of strong types can aid such implementations. [8]

(c) The Rust programming language provides `unsafe` blocks, that allow certain aspects of the type system to be circumvented. One of these aspects is the ability to dereference raw pointers, that can provide unsafe memory access. Discuss, with examples, why Rust provides for unsafe memory access in this way, and whether the presence of this feature indicates a problem with the language. [7]

3. **(a)** Over the past decade, or so, processors have moved from emphasising single threaded performance to supporting large numbers of concurrent threads, primarily by moving to multicore processor designs. Explain why this transition has occurred. [4]

(b) Message passing systems are often said to avoid many of the problems inherent in lock-based concurrency. However, message passing systems can still deadlock and race conditions can still occur. Explain how deadlocks and race conditions can occur in message passing systems and discuss whether you think they are more or less likely to occur than in systems using lock-based synchronisation, justifying your answer. [5]

(c) Erlang is a message-passing concurrent language that adopts a "let-it-crash" approach to error handling, where responsibility for handling failures in a task is given to a separate supervisor task. Discuss whether the let-it-crash model is an appropriate way of providing robustness in a massively concurrent system, or if in-process error detection and recovery is

more suitable. Your answer should include an explicit discussion of the trade-offs between the two approaches. [6]

(d) One way of implementing asynchronous I/O operations is to structure the code as a set of coroutines. In this model, operations that may block return a `Future<T>` representing a value of type T that will become available at some later time. The runtime system for such languages includes an `await` operation, that allows a function to wait for a result to become available without blocking the thread, by passing control to other coroutines while waiting. Discuss whether this is a good approach to writing asynchronous code. Explain what are the strengths and weaknesses of this approach to structuring code. [5]

4. **(a)** This course has argued that the way we write systems programs has to change if we are to successfully develop secure, highly concurrent, and networked systems in the future. In particular, it has been suggested that we should use strongly typed, memory safe, programming languages to improve overall robustness of the code, that those languages need to give control over data layout and memory access, and that we need to pay special attention to input parsing and handling of untrusted data to ensure system security.

Do you agree? Discuss the extent to which you believe that changing the programming language, and using better tools to parse and process untrusted input, will help to address the challenges inherent in building secure, highly concurrent, networked systems. Discuss which programming language and runtime features, and what features of parsing tools, you consider important to support secure systems programming, and which feature are harmful or not useful. Give examples to illustrate key points. [20]