University of Glasgow

Thursday, 4 May 2023
14:00-15:30 BST
Duration: 1 hour 30 minutes
Timed exam – fixed start time

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

ADVANCED SYSTEMS PROGRAMMING COMPSCI 5083

Answer question 4 and any two other questions

This examination paper is an open book, online assessment and is worth a total of 60 marks.

1. **(a)** In the early 2000s, processor design moved from emphasising single threaded performance to emphasising support for parallelism. This led to the number of processors cores per device increasing rapidly. Explain why this transition occurred. [5]

(b) The Rust programming language expresses the concept of ownership, and the ability to transfer ownership of data, as part of its type system. This allows Rust programs to safely pass mutable data between threads without race conditions. Other programming languages, for example Erlang, avoid race conditions by only allowing immutable data to be passed between threads. Discuss whether you think the benefit of being able to safely pass mutable data between threads in Rust is worth the complexity introduced in the type system. [5]

(c) The Erlang programming language adopts the "let-it-crash" approach to error handling, where responsibility for handling failures in a task is pushed out to a separate supervisor task. Discuss whether this approach to error handling is appropriate for use in systems programs, or if in-process error detection and recovery is better suited to handling errors in systems programs. Your answer should include an explicit discussion of the trade-offs between the two approaches. [10]

2. **(a)** A common pattern in some C programs that process binary data, for example network packet formats or compressed image or video file formats, is to write code similar to the following example:

```
struct rtp_packet {
  unsigned short   v:2;   /* packet type             */
  unsigned short   p:1;   /* padding flag           */
  unsigned short   x:1;   /* header extension flag  */
  unsigned short   cc:4;  /* CSRC count             */
  unsigned short   m:1;   /* marker bit             */
  unsigned short   pt:7;  /* payload type           */
  uint16_t         seq;   /* sequence number        */
  uint32_t         ts;    /* timestamp              */
  uint32_t         ssrc;  /* synchronisation soource */
}

...

char *buffer = malloc(BUFLEN);
if (recvfrom(fd, buffer, BUFLEN, 0, &addr, &addrlen) > 0) {
  struct rtp_packet *pkt = (struct rtp_packet *) buffer;
  if (pkt->v == 2) {
    // Process packet
    ...
  } else {
    ...
  }
}
```

This example uses `recvfrom()` to read data from a UDP socket and stores it in `buffer`, a heap allocated array of bytes. It then takes a pointer to a structure of some different type,

in this case `struct rtp_packet`, and uses a cast to make it point to the contents of the buffer, allowing access as if the buffer was of that type.

Discuss what are the advantages and disadvantages of this approach, and state whether you think it is an appropriate design pattern to use in modern systems. Your answer should mention the type and memory safety implications of this technique. [10]

(b) It has been claimed that the use of modern programming languages, with expressive type systems, can improve software security. Do you agree? Discuss and justify your answer, giving examples to illustrate key points. [10]

3. (a) One of the key features of systems programming languages is that they provide control over the layout of data in memory. Explain why this is important, and outline what features of the C and Rust programming languages provide this control. [10]

(b) Rust distinguishes between shared immutable references to data (`&`) and uniquely owned mutable references (`&mut`). Explain why this distinction is made, and what benefits it provides. [4]

(c) The design of Rust encourages programming in a largely functional style, passing immutable data and writing pure functions. Discuss whether you think a functional programming style is appropriate for systems programming, highlighting both advantages and disadvantages. [6]

4. (a) The recommended reading included Shapiro's paper entitled "Programming language challenges in systems codes: why systems programmers still use C, and what to do about it" (ACM Workshop on Programming Languages and Operating Systems, San Jose, CA, USA, October 2006) and "The bugs we have to kill" by Bratus, Patterson, and Shubina (USENIX ;login:, August 2015). These papers suggest 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, the authors suggest 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.

Do you agree with the thesis of these papers? 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 are harmful. Give examples to illustrate key points. [20]

END OF QUESTION PAPER