CSE 221 Homework 2

October 31, 2008

Due: Thursday, November 6 at 11:00 am

Submit your solutions as a hard copy at the beginning of class or e-mail them as a PDF to ebuchana@cs.ucsd.edu *before class starts* at 11:00 am. Late homeworks will *not* be accepted.

Answer each question completely, and support your answers with material from the papers and your own critical arguments, as appropriate.

1. Virtual Memory

   (a) Describe how virtual address translation works on a modern architecture using a hardware-managed translation lookaside buffer (TLB) and a monolithic operating system (e.g., Linux or Windows). Imagine that we are running Emacs as an application on the system. Start with the step where Emacs executes an instruction that makes a memory reference to a page that has been paged out to disk, and end with the step where the Emacs instruction finishes execution. It is sufficient to simply list the steps, a detailed discussion is not necessary.

   For example:

   Step 1) Emacs issues an instruction that accesses a virtual address on a page that is not in TLB, or whose TLB entry has been marked invalid.

   (a) A number of different operating system structures have been proposed over time, including how virtual memory is managed. Again consider the case where Emacs is running as an application on a system and causes a page fault. Choose three of the following four systems:

   - L4
   - Xen
   - VM/370
   - Mach

   For each of the systems you choose, describe how they handle a page fault by answering the following questions:

   1. What part of the system (whose address space) receives the initial page fault exception?
   2. What part of the system (whose address space) stores and manages application page tables?
   3. What mechanism does the system use to vector a page fault from (a) to (b)?

2. Operating system dualities

   (a) Which of the systems we've read are message-oriented? Which are procedure-oriented? Select one of each, and briefly describe what constraints made the authors choose the orientation of their system. Were any arguments for the decision used that Lauer and Needham would consider invalid?

   (b) Is UNIX message-oriented or procedure-oriented? What are some of the services and interfaces that follow from its orientation, and what might those look like if UNIX had been designed with the other orientation?

3. RPC Optimization

   System designers often choose to implement important features as special cases of more general mechanisms to improve performance. On the other hand, special-casing can lead to inconsistent interfaces and complex implementations.

   (b) Briefly describe the optimizations made in the Remote Procedure Calls paper to make RPC calls fast. Did the performance improvement achieved by these justify the added complexity?

   (c) What further optimizations did the V Kernel system designers implement? Were they worth it?