Course Development Project – Final Report

Winter 2018

NAME

Course Background

In this course students learn about the elements of Computer Systems, including processes and process control, virtual memory, concurrency, network programming, and parallel programming. Most of the students taking this course are juniors and seniors. However, the programming concepts that most have learned to this point don't involve such low-level interaction with the operating system kernel, so the course presents a new paradigm and environment to which most are not very familiar. Many are used to developing with graphical tools and are less familiar with command-line use. Also, the asynchronous approach to many of the programs that are being designed make it non-intuitive in the more common procedural programming paradigm.

I have worked with other professors in the creation and development of this course, including one who teaches the same course in odd semesters and two who trade off teaching the prerequisite systems course. The course material is based largely off content developed by CMU professors, who wrote the textbook, though it has been adapted to BYU curriculum.

Learning Outcomes

- Linux Systems Programming
    - Students will demonstrate mastery of Linux system programming.
    - Students will be able to use Linux systems calls that are defined in C from within a C++ program to control processes and threads and to use shared memory synchronization, polling, file I/O, and sockets.
    - Students will be comfortable using the Linux command line and development environment to compile programs with Makefiles and for navigation, job control, I/O redirection, utility commands and manual pages.
    - Students will gain experience with advanced C topics (memory management, pointer manipulation, strings, etc.)
- Multiprocessing, Concurrency, and Exceptional Control Flow
    - Students will demonstrate mastery of multiprocessing, concurrency mechanisms, and exceptional flow control.
    - Students will write concurrent programs that provide efficiency gains from multiprocessing and demonstrate proficiency in threads, processes, mutual exclusion primitives, deadlock avoidance, signal handling and exceptional flow control.
    - Students will understand how the operating system implements processes, threads, concurrency mechanisms, including how the operating system provides scheduling.
    - Students will understand signal handling, exceptional flow control and virtualization.

- Virtual Memory
    - Students will demonstrate how virtual memory works and how it can affect program performance
- Internet Programming
    - Students will demonstrate a mastery of Internet programming, including using sockets, protocols, and non-blocking I/O.
    - Students use the BSD socket API to implement client-server programs using non-blocking I/O and polling to scalably multiplex large numbers of sockets.
    - Students use effective testing techniques for networked programs, including stress testing, load testing, and fuzz testing.
    - Students understand the role of TCP, IP, HTTP, DNS, and TLS as they relate to Internet services.

The learning outcomes embody concepts that are at the very heart of computer systems, and by learning these concepts, the students have a better understanding of not only programming but the systems on which programs run, how to most efficiently use those systems as both a user (i.e., of the command-line) and a developer.

Course Activities
The graded learning activities consist primarily of 1) homework assignments and 2) labs. The homework assignments are generally hands-on assignments, designed to guide a student through a given topic, either to program concepts that we have learned in class or to otherwise analyze and build on programs already written for the purpose of learning the concepts. Sometimes the homework assignments consist of problems from the textbook, designed to help them think about and solidify the concepts. The labs are larger projects in which the students implement concepts in a programming assignment, where they have a specification and are expected to produce a program that matches. The homework assignments are intended to help prepare the students for the labs and for learning assessments. The idea is to guide and teach students with the homework, lectures, and textbook reading, and then let them gain a deeper understanding of the concepts by applying them in the labs.

The labs and homework assignments are supplemented by in-class discussions and group activities. The former often occur at the beginning of my class. I post questions on the screen, and I ask the students to discuss the topic with their neighbors and answer them. The group activities were introduced later in the semester. I noticed students struggling with concepts and learning activities, and they could have benefitted from interactions with others. Ultimately, facilitating group discussion and conceptualization was an attempt to help the better performing students to help the students who were struggling.

Assessments of Student Learning
The course provides three major assessments of learning: two midterm exams and a final exam. The midterm exams cover approximately 75% of course material, and the final exam is comprehensive. The material on the exams is mostly based on concepts that they will have learned through the labs and homework assignments. The idea is that a student who has

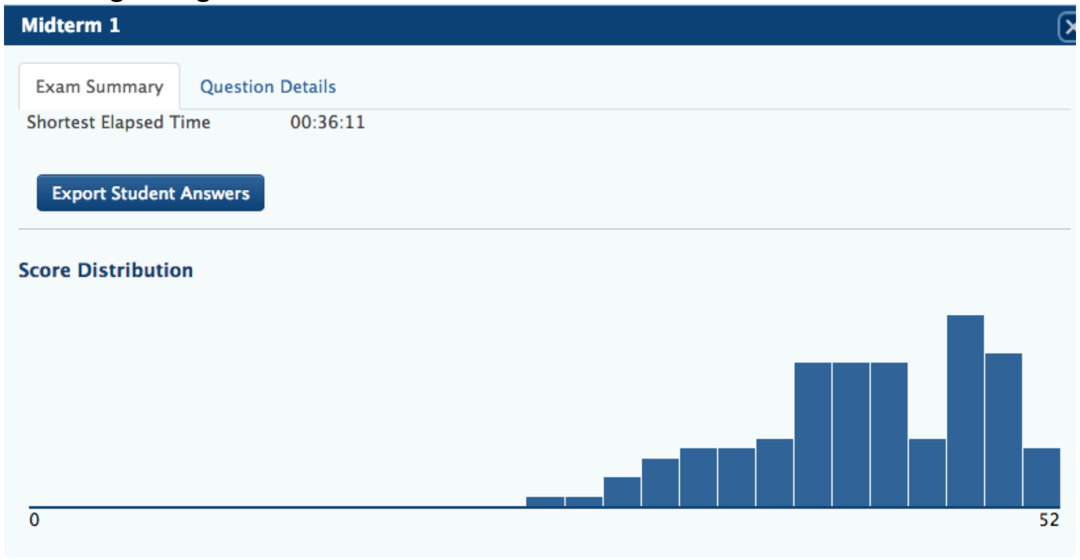
completed the labs and homework assignments will at the very least have been exposed to and, in the best case, have acquired the knowledge not only to complete the learning activities but to truly learn the concepts in the process. The scores, therefore, will typically be a reflection of the knowledge acquired through the learning activities. And because the learning activities are focused on helping the students acquire the learning outcomes, it follows that the students that do well on the exams are internalizing the learning outcomes.

The format of the test was mostly True/False or Multiple Choice, with about 25% being short answer. The tricky parts about that are that 1) it's difficult—if not impossible—to give partial credit for incorrect answers; and relatedly 2) it's takes some effort to build large, multi-part questions whose overall value is appropriate for the material they cover.

Student Achievement of Learning Outcomes
The completion of the learning activities and the assessment of the students on the learning outcomes has demonstrated that students are achieving the course learning outcomes. At this point in the semester, the only midterm that has been administered covers only a subset of the learning outcomes of the course, but I show evidence below that students are demonstrating achievement of the learning outcomes.

The first midterm covered some of the following learning outcomes: **Linux Systems Programming**; **Multiprocessing, Concurrency, and Exceptional Control Flow**; and **Virtual Memory**.

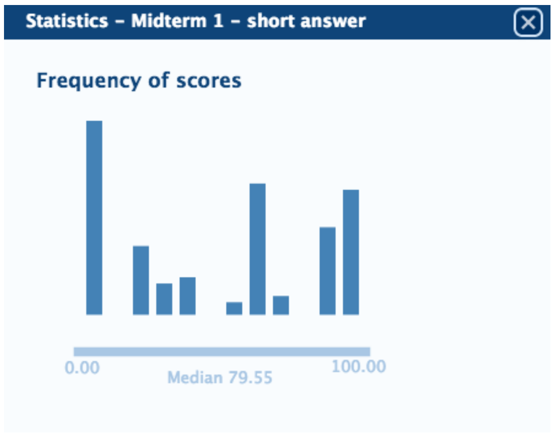
The distribution of combined scores on the first two of these learning outcomes is shown in the following histogram:



Shortest Elapsed Time 00:36:11

It appears from the scores that most students are getting a grasp of the concepts in the first and second learning outcomes.

Scores for the last topic, Virtual Memory, is shown in the following histogram:



Clearly there is work to be done to help the students with this learning outcome, as the assessment indicates performance that is less than expected. I also include in the appendix some students comments from a midcourse evaluation that indicates some success in this area.

Steps Planned or Taken to Improve Teaching and Student Learning

Many students have met with me in my office or have expressed through midcourse evaluations that the biggest challenges with this particular course are related to the concepts of the C programming language, Linux command-line usage, and memory management. My teaching colleagues, those teaching the same course in off semesters, and those teaching the prerequisite sister course, concur with that analysis.

In order to improve my teaching and the student learning in my classroom, one major change I intend to implement in future semesters is to have the learning activities at the beginning of the course include more introduction to foundational concepts that challenge students throughout the course. I specifically refer to the challenging concepts mentioned previously—C programming language concepts, Linux command-line usage, and memory management. Many students have met with me in my office or have expressed through midcourse evaluations that those are the biggest challenges with this particular course. The change to the activities at the beginning of the course will better help students with these challenges by allowing them to see the concepts hands-on and being a resource to refer back to for later learning activities.

I also implemented two changes mid-semester. The first, mentioned previously in this report, is the group activities. I observed that many students were struggling in part because they weren't interacting with other peers in the class to learn and apply the concepts. For the last three labs I am requiring group discussion and conceptualization, in an attempt to have the students that are better grasping the concepts to help the students that are struggling. But more generally, the goal has been to get students to work with others. I have already received anecdotal feedback that the change has been an effective one for the students.