Systems Area CQI Report

Systems Area CQI Committee (Moore, Freudenthal, Tosh, Ward)
September 28, 2021

The Systems Area encompasses teaching of concepts and skills related to how computer systems work, focusing on the hardware-software interface. The courses address how properties of modern hardware (e.g., processors, networks, storage hierarchies) influence the design and implementation of systems software.

Systems Area courses include three required courses – CS 3432 (4 credit hours, currently titled Computer Architecture I – Basic Computer Organization and Design), CS 4375 (3 credit hours, currently titled Theory of Operating Systems), CS 4175 (1 credit hour, titled Parallel Computing) and one technical elective – CS 4316 (3 credit hours, titled Computer Networks).

The CQI Committee met on August 16, 2021, and discussed the following topics:

1. Proposed changes to systems curriculum
2. Approach to revised learning outcomes
3. Spring 2021 course reports

Ward proposed that the title and description for CS 4375 be changed to reflect the content and goals of the course, and that the learning outcomes be revised to be assessable, understandable to students, able to be taught, and of a reasonable number. Details of Ward's proposal, as revised following the committee discussion, are in the accompanying CS 4375 Proposed Changes document.

Freudenthal proposed that the title of CS 3432 be changed to Computer Organization since as a first systems course, it is not an architecture course. The committee concurred with this change, and it is included in the accompanying CS 3432 Proposed Changes document.

Freudenthal proposed that CS 4375 be moved to 3000-level since operating systems concepts are important for other 4000-level courses, such as Computer Networks and Computer Security. Currently, students often wait to take CS 4375 until their last year or semester. We did not discuss specific changes to prerequisites for other courses, but moving CS 4375 to 3000-level would allow it to be made a prerequisite for other 4000 level courses. The committee met again on September 28 and concurred with this change, and it is included in the accompanying CS 3432 Proposed Changes document.

Freudenthal proposed that CS 3432 be changed to a 2000-level course that could also be taught at EPCC. The rationale for moving it to 2000-level is that computer organization concepts are important for other 3000-level courses (such as Programming Languages) but students often take CS 3432 after those courses. The committee suggests that the Systems committee and the CS Department discuss the implications of this change over the next year.

A major goal of the committee was to revise the learning outcomes for CS 3432 and CS 4375. Criticisms of the current outcomes were that there are too many to teach and assess, some are not directly assessable, some do not enable students to understand what is expected, and some are

redundant or no longer needed. In particular, the required outcomes mapped to systems course for Cybersecurity Certification have changed and are no longer as detailed. Tosh provided the revised Cybersecurity outcomes/topics that need to be mapped. The committee decided on an approach to revising the learning outcomes with the goals of making them assessable, understandable by students, and at a high enough level of abstraction to produce a reasonable number while leaving some flexibility for adapting to changes in the systems area and for accommodating different teaching and assessment practices.

Ward presented the CS 4375 course report to the committee. The report is based on Ward's Spring 2021 course and includes detailed assessment results for the current learning outcomes. Ward also presented proposed new outcomes for CS 4375. The committee discussed the proposed new outcomes in detail and made suggestions. Ward revised the proposal accordingly. The proposed new outcomes are in the accompanying CS 4375 Proposed Changes document.

Freudenthal presented the CS 3432 course report to the committee. The report is based on Freudenthal and Pruitt's Summer 2021 courses and includes detailed assessment results for the current learning outcomes.

The committee made suggestions for revising the CS 3432 outcomes to be directly assessable and less detailed and redundant. Moore and Freudenthal worked together on the proposed new set of outcomes. The committee met again on September 28 and approved the proposed new outcomes. The proposed new outcomes are in the accompanying CS 3432 Proposed Changes document.

A course report has not been produced for CS 4175 (Parallel Computing) because it was not taught during Spring 2021.

CS 3432 Proposed Changes

Systems Area CQI Committee Meeting (Moore, Freudenthal, Tosh, Ward),
September 28, 2021

Change 1: Rename as Computer Organization

Rationale: The course is not a computer architecture course, so the current name, "Computer Architecture I: Basic Computer Organization and Design", is not appropriate. The course is a first systems course about how computer systems are organized in terms of the hardware-software interface. We propose renaming the course simply "Computer Organization."

Change 2: Revise the course outcomes

Proposed new outcomes:

Subject Areas (and abbreviations):

- Hardware/software interface (HSI)
- Architecture (A)
- Numeric Representation (NR)
- Linearization (L)
- Tools (T)

Level 1

HSI-1. Define and explain the purpose of an instruction set architecture (ISA).
HSI-2. Explain the relationship and differences between a high-level programming language, assembly language, and machine language.
HSI-3. Describe the fetch-execute cycle in terms of the hardware-software interface between machine instructions and processor components.
NR-1. Explain the relationship between a high-level language basic data type (e.g., signed or unsigned integer, floating point number) and its representation as a bit pattern inside the computer.
A-1. Describe the basic components of a processor (e.g., register file, special-purpose registers, control unit, memory) and how they interact with one another.
HSI-4. Explain how procedures are supported by processor hardware.
HSI-5. Explain exception/interrupt handling in terms of the hardware-software interface.
HSI-6. Explain various ways an operand can be addressed in an assembly language instruction.
HSI-7. Describe the process of compiling/assembling, linking, loading, and executing a program.

Level 2

NR-2. Convert between different integer data representations (e.g., decimal, binary, hexadecimal, octal).
NR-3. Interpret the bit representation of a floating-point number.

NR-4. Perform addition and subtraction on two's complement representation of integers.
NR-5. Use bitwise operators to access and manipulate values stored in a subset of bits within a byte or word.
NR-6. Determine range and precision (if applicable) of numbers that can be stored for a given data type and determine whether an integer operation will result in overflow.
HSI-8. Convert between machine and assembly language representations of instructions – i.e., encode and decode instructions.
HSI-9. Trace the datapath through the processor for a given class of instructions (e.g., arithmetic-logical, memory access, conditional branch)
HSI-10. Trace the execution of an assembly language program with procedure calls in terms of allocation and deallocation of stack frames and register and memory contents.
L-1. Translate expressions and assignment statements from C to assembly language.
L-2. Translate Boolean logic and control flow constructs (decisions, loops) from C to assembly language.
L-3. Translate operations on arrays, structs, and pointers from C to assembly language.

Level 3

HSI-11. Implement/debug simple imperative programs in assembly/machine language.
HSI-12. Write or call a procedure with local variables, parameters, and return value in assembly language.
HSI-13. Implement a simple interrupt or exception handler.
T-1. Compose, compile/assemble, execute, and debug simple programs in a command-line environment, using appropriate modularization and multiple files.

CS 3432 "Arch1" CQI Report for pandemic semesters (data collected during S21 Semester)

Eric Freudenthal
14 August 2021

Arch 1 primarily introduces students to tools used for systems programming and the key concepts underlying the dominant Von Neumann architecture in the context of programming in C, assembly, and machine language. In this class, students construct programs that implement foundational pointer-based data structures that execute under POSIX and communicate with (simple) memory-mapped i/o devices, respond to interrupts, and enter/exit low power sleep modes. These programs are developed using widely-used command-line development tools that are introduced to students during this course included curses-based editor, compiler, build system, repository, bash, etc.

Due to the COVID-19 pandemic's onset mid-S20, with few exceptions, all courses were taught online until f21. Arch1 lecture and lab sessions have been taught in a synchronous online manner. This report primarily describes adaptations to and observations of the synchronous sections taught by the author of this report.

Students were encouraged to assist each other in determining and validating solution strategies during all learning activities including lab and (formative) assessment exercises. Summative assessment during these semesters in my sections was primarily through Socratic questioning during bi-weekly 1-on-1 coaching sessions with members of the teaching team who updated spreadsheets entries corresponding to assessed skills for each student.

At the end of S20, the teaching team observed that many students who failed to understand advanced concepts such as interrupts and function calls also failed to comprehend the fundamental concepts underlying the related in the fetch-execute cycle and the program counter's (PC's) central function and role in a single-issue processor, even during sequential execution. In subsequent semesters, substantial teaching time including learning/assessment activities were focused on examining the PC's central role in the identification of the next instruction to fetch/execute, and why/how it is updated during execution and assessment was at 3 (Analyze).

At the end of S21, the grade distribution was

- 36% A
- 8% B
- 8% C
- 4% D
- 18% incomplete (mostly pandemic-related)
- 16% drop (mostly pandemic-related)
- 18% F

Thus, only 53% of 49 enrolled students earned passing grades of C or better by the end of the term. An additional ~10% (5) of students completed the course during Summer 2021. Several students who were assigned failing grades indicated that their (sustained) enrollment was motivated by minimum course-load requirements stipulated by financial aid programs.

Assessment

When learning outcomes were grouped into "families" for the 2019 CQI cycle, the intention was to identify themes to be taught/assessed with anticipated prerequisite knowledge and outcomes serving as guidelines. While the teaching team was guided to monitor and guide student mastery related to all families, no data was collected related to written communication or details of numerical representation. Furthermore, as described, below, timing-related outcomes taught and examined differ from those enumerated in the "timing" family.

The 'timing' family of detailed learning outcomes from the 2019 CQI focuses on instruction timing. The as-taught course instead primarily focused on counter-timers and the use periodic interrupts that are counted to measure time and schedule events. This time-keeping idiom introduces techniques used in operating system and networking code to track the passage of time. The committee should discuss this change of focus.

Fractions (expressed as percentages) are of students who earned a passing grade of C or better by the end of S21. Fractions below the department's minimum threshold of 70% are indicated in **boldface.**

Learning outcomes systematically observed during coaching sessions and discussed by the teaching team.

- Level 3 (analyze, synthesize)
  - (GA, was level 1) Can describe and analyze the interaction of program counter, stack pointer, and status register in the context of sequential & branching execution, function calls (68%), precise interrupts (80%) that conditionally awaken an interrupted program from a low-power sleep state (80%)
  - (NR) Can interpret and analyze difference between values represented as signed (2's complement) and/or unsigned integers including comparison (96%) and selection of relevant flags **(68%),** can choose appropriate representation for context (e.g. hex, signed, unsigned, octal) *not tabulated.*
  - (MP) Can synthesize and debug programs in C and assembly with appropriate modularization (100%) and symbol names (100%).
  - (Dev) Can synthesize programs that interact with (simple) memory-mapped i/o devices and respond to interrupts they generate. (82%)
  - (WC) Written communication, members of the teaching team were attentive to and informed students when their programs were difficult to understand. not tabulated.
  - (Timing) Use of periodic timers to implement delays in state-machine driven programs. Not tabulated, but likely 100% because this was an enforced requirement of non-optional lab assignments.
- Level 2 (application)
  - (Subroutine, GA) Can implement subroutine linkage for an imperative language (**58%**) including automatic variable allocation **(54%)** in assembly language.
  - (Tools)* Can productively use linux command-line tools including editor, compiler, linker, make, and a source-code repository in the construction of programs composed from multiple source files (all 100%).

- (NR) Can convert between rational and single precision floating point representations, can determine of maximum and minimum representable floating point values from first principles. *not tabulated*
- (GA) Implement (apply) type conversion and analyze contexts where conversion will yield incorrect values. *not tabulated*
- Can utilize a periodic interrupt to implement timers that affect program behavior. (100%)
- (L) Can convert infix arithmetic expressions (91%) including pointer arithmetic & array stride (96%), block-structured control flow (95%), and switch statements as branch tables (70%) into linearized structures suitable for encoding in assembly/machine language.
- (Dev) Can describe gross characteristics and determine suitability of various types of volatile and persistent storage devices for common data-storage applications. *not tabulated*
- (GA, MP) Can use bitwise operators to access and manipulate values stored in a subset of bits within a byte or word. (91%)
- (GA, NR) Can encode/decode machine language. *not tabulated*

Revisions during subsequent SU21 semester taught by Eric Freudenthal with David Pruitt (TA)

- The central role of the PC (and eventually SP & SR) in the context of sequential instruction and later branching, function calls, and interrupts was stressed heavily and described throughout lectures and coaching session as the course's primary learning outcome. In that semester, the fraction of students who earned a passing grade who were observed to demonstrate mastery of topics related to subroutine linkage was dramatically higher:
  - Return value: 90%
  - Parameter passing: 83%
  - Allocation of auto (a.k.a. local) variables: 83%
- The instructor led many lab sessions and observed that advanced software engineering strategies embedded within demonstration code obfuscated low level algorithms and techniques central to the course. Simplified alternatives were created that students appeared to find far more accessible.

In the past, all outcomes not tabulated during 1:1 coaching sessions have be easily measured and tabulated using conventional assessment instruments employed during in-person classes.

Official CS3432 Learning Outcomes (adopted 2019)

family	**Prerequisite knowledge from CS1,CS2, digital design, discrete, and precalc**	**Students will be familiar with...**	**Students will be able to effectively apply skills...**	**Students will be able to analyze & synthesize solutions..**

NR: numeric representation & ops	- familiar with radixes, signed representations, and scientific notation		- convert hex, dec, signed-dec, binary binary metric - signed/unsigned comparison (flag: **68%**, order 96%) - Add-with-carry - cast/sign-extend - floating point	- can determine appropriate representations for elementary types and design low-level programs that compute arithmetic results
L: linearization	- algebra, - arithmetic & control-flow structures of an oo language - block structures,		- expressions (incl side effects) 91% - control flow (if/while/for) 95% - translate boolean logic - branch tables **70%** - op on arrays, structs, and pointers 96%	- can translate infix expressions and block-structured programming constructs to assembly language
GA: gross architecture	- able to program in at least one oo language - familiar with cobinational and sequential logic	Can describe the fetch-execute cycle in the context of the roles of PC, SP, flags, registers and memory.	- select appropriate instructions - specify operand order - encode and decode instructions - specify addressing mode - utilize interrupt mechanism - implement interrupt handlers - implement sleep/wakeup	- can implement and debug simple imperative programs in assembly or machine language *100%*
Ti: timing	- algebra - synchronous logic - frequency		- determine cycles/instruction - determine which instructions repeat in a loop	- can compute the execution time of a simple loop - can design a loop that delays a specified amount of time
Sub: subroutine linkage & separate compilation	- in oo languages studied in CS1/2		- parameter passing - return value - allocation of auto vars - register usage - global/local symbols	- can write or call a method with local variables, parameters, and return value in assy lang

VA: variable allocation	- in oo languages studied in CS1/2		Can define and use variables with various.. - scope: visibility (file/method/program) - variable lifetime (program/method) - size - alignment - arrays - pointers - structs	Can appropriately allocate static and auto variables including arrays and pointers in assembly language
T: tools	- ide - hierarchical filesystems		Can effectively employ in the composition and debugging of programs - editor - compiler - make - bash - gdb - source code repo	Can compose and debug simple programs in a command-line environment (100%)
WC: written communica-tion	- proficient in English		Can - interpret technical documentation on familiar topics - describe implementations that they design - recognize/use technical terminology - appropriate documentation for code	Can appropriately document simple programs
MP: mature programm-ing	- proficient in OO programming - appropriate comments - can modularize - appropriate symbol names - coding style		Can utilize in a program - appropriate comments - modularize 100% - imperative programming - appropriate symbol names 100% - coding styles 100%	Can appropriately modularize and document simple programs consisting of multiple files

Perf: advanced topics for performance		- pipelining - vectoriza-tion - predicated instructions		Can identify when these topics are relevant to constructing an efficient solution.
DEV: devices	gates, latches, (de)multiplexers, ALUs, switches, counters	- gross characteriscs of memory & storage devices - counter-timer	can implement - simple programmed i/o - interrupt handlers	Can design programs that implement simple programmed i/o and interrupt handling - can determine the types of storage devices suitable for a variety of uses.

Recommendations: The systems committee should discuss:

- Whether/how "timing" learning outcomes should be modified to include/focus on the use of periodic interrupts to schedule events.
- Potential conversion of 3432 to a 2xxx course to be generally attended concurrently with 2302 and likely also taught by EPCC.
- Now that we have multiple instructors for this course using multiple instructional approaches, whether/how we might collaborate in identifying and sharing best practices.

CS 4375 CQI Report, Spring 2021, Section 2

Nigel Ward, May 19, 2021

Course Title and Description: **Theory of Operating Systems**

Process and thread management, concurrency, memory management, processor scheduling, I/O management and disk scheduling, and file management

Recommendations

Recommendation 1: Make this an in-person course again.

An online course is intrinsically inefficient and inferior in many ways. due to the time lost waiting for students to turn on their microphones, the lack of high-bandwidth exchange of ideas in groupwork exercises, and so on. It also appeared to impact attendance. An in-person course would likely improve both the quantity and depth of learning.

Recommendation 2: Change the course title to Systems 1.

This title is obsolete. Something needs to be done to indicate that this course is not a theory course, and that it includes the basic concepts of networking. Eric's proposal was, I believe, Systems 1. An alternative would be Operating Systems and Network Concepts.

Recommendation 3: Update the course description.

Details: This should include "interprocess communication" and "networking". "File management" should be simply "file systems". "Disk scheduling" should be omitted, since while it provides interesting case studies in algorithms and data structures, introduces no new concepts and it is seldom important for most purposes. Networking should be mentioned, and also security implications.

Recommendation 4: Update the course outcomes.

The 2018 outcomes are problematic in that:

- there are too many to teach
- some are aspirational mentions of advanced topics that cannot be covered in a first systems course.
- there are too many to assess (50), including many which are redundantly stated at different levels
- many are not directly assessable, being instead just mentions of important concepts, or of technologies that illustrate those concepts, or of interesting juxtapositions or abstractions that relate to deeper understandings of those concepts
- many are worded in ways that do not enable students to understand what they mean or what is expected.

Some recommendations addressing some of these issues were made in the Fall 2019 Systems Committee CQI Report, but apparently never acted upon.

Appendix B a draft new list of outcomes, and Appendix C is an itemized critique of the current outcomes.

Recommendation 5: Cap enrollment at 40.

With 40 students I would have been able to call on every student at least once, most days. With 40 students the TA would have been able to provide more timely and individualized feedback. With 40 students we would have relied less on multiple-choice questions, giving the students more practice in clear technical writing. With 40 students we would have had the time to meet with each student for at least one code demo. Overall, in a small class there would have been better student motivation and more learning.

Recommendation 6: Give better readings on networking and security.

My hope was that with three decent resources, students would be able to synthesize to come up with their own understanding. This did not seem like an unreasonable expectation for a senior class, but many students did not reach this level.

Recommendation 7: Give the students worksheets on delay, latency, and powers-of-two computations.

My belief was that students would have the basic mathematical competencies to set up simple equations from simple story problems, but for most students this was not true. Clearly these need reinforcing.

Recommendation 8: Generally continue to monitor and improve.

It is possible that some of the other shortcomings were transient effects of teaching online. In any case, instructors in future semesters should seek better ways to teach and reinforce the other outcomes that were unsatisfactory, and monitor their effectiveness in the next CQI cycle.

Appendix A: About the Course

Teaching Format

The course was taught in Blackboard, synchronously, but recorded. Class time was devoted to answering student questions (about 10%), lecture (25%), having students answer questions or work through problems (50%, often in breakout groups), and going over homework assignments and quiz and test questions (15%).

Office hours were held in Blackboard. As is typical for courses at this level, few students took advantage of these. As a consequence, the TA's contributions were mostly in grading and in visiting groups during the breakout sessions.

Review sessions were held in Zoom, with two duplicate-content in-person session.

Student Background

Most students in the class are juniors, with only about 25% taking it in their final semester. About 10% had previously taken Networks, with another 10% taking it concurrently.

Conversely, of the students in Networks this semester, Deepak reported that 47% had already taken OS and 50% were taking it concurrently. Thus the idea of making 4375 a prerequisite to Networks, as occasionally discussed, seems feasible.

Student Engagement

Attendance dropped from 100% the first week to about 80% most days. This I believe is anomalous and probably mostly due to the online format.

Those attending were generally engaged and when called on were able to answer questions or at least say something pertinent, with a few exceptions. About 5% of students did not respond when called on, with "microphone trouble" or "just stepped away" were the most common reasons/excuses. In groupwork on average about 90% of the students present appeared to be actively participating.

Student feedback, in the Week 11 survey, confirmed my interpretation that for many students engagement was good. Two typical comments were:

> I think so far the class is going well. I still enjoy the way the class session is structured. I think it is nice as it keeps me engaged and not bored and I am actually paying attention to the class. I like the assignments we have done so far, it's good practice and it helps me understand the topic we are covering better...
>
> To be honest, I actually kind of enjoyed this class. I came in with the idea that it wasn't going to be a good class since everyone talks about how challenging it is and how much they suffered when they took it. I heard all these horror stories about OS and I was pleasantly surprised with what it turned out to be. That's not to say that it isn't a challenging class because it takes a good deal of time and effort to do well but I found I enjoyed a lot of the assignments and learned/reinforced a lot of very interesting concepts. ...

Textbook and Resources

Operating Systems: Three Easy Pieces, by Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau, a free online textbook, with print copies available, was the primary textbook. This worked well.

Introduction to Computer Networks, by Peter Dordal, was the secondary textbook. This was also free online. Its coverage was fine for IP, DNS, and layers, but not for security. As a supplement I assigned some sections of Kurose & Ross Chapters 1 and 2, but this was not a great solution because, although pdfs are available, their copyright status was not clear, and because it's the textbook for Networking.

In addition I assigned a number of videos, mostly because animated descriptions of algorithms and processes, while possible on a whiteboard, are difficult to do well in Blackboard.

Assessment

Students took quizzes on the reading, due 5 minutes before class almost every day. These were mostly simple questions and were more for motivation than assessment purposes.

Assignments were occasionally assigned to be done in pairs, but these assignments were not a major part of the grade or the outcomes assessments.

Tests were given synchronously, rigidly timed, and mostly done in lockdown browser with video surveillance, although we never looked at the videos. The instructor and TA were vigilant for issues, but found no evidence of plagiarism or cheating on the assignments or tests.

Grade Distribution

49 students were enrolled on the first day of classes. Two dropped, and the one who told me why said that he couldn't find the time, with other classes and work. One had Covid before the second test, and did nothing after that point. Thus 46 took the final examination, although one did not submit part 2; he failed and had been long doing poorly. The other had stopped attending class for two months, for financial reasons. Three students earned Ds, one had stopped attending due to a work, another stopped attending due to "personal matters", and the other stopped attending and did not respond to inquiries. In sum

A	7	14%
B	21	43%
C	13	27%
D	3	6%
F	3	6%
W	2	4%
total	49	

In only one case did it seem that there was a chance of inadequate preparation being a significant factor. I sent the name of the student in question to the instructor of the prerequisite class, hinting that he might consider the situation and whether this might suggest tightening criteria or other adjustments to reduce the risk in future of students taking OS only to fail.

Appendix B: Assessment of Outcomes with Respect to the Proposed New Set

B1: Choice of Outcomes to Assess

After long discussions with Eric, I was able to determine the underlying intent behind each of 2018 outcomes and understand which matter most and how they relate to the overarching learning goals for the course, which I summarize as:

> Learn concepts that will be foundational for further study, whether course-based or on-the-job, of:
>
> 1. Computer Security and Computer Forensics
> 2. Systems Administration and Network Administration
> 3. Systems Programming and Network Programming, including developing for non-standard platforms, such as embedded systems, systems in the cloud, and high-performance (parallel) systems

4. Effective use of modern computers, notably Windows and Posix systems

I then worked to restate each essential learning outcome (marked * in Appendix C) in a way both assessable and comprehensible to students. The result, below, is my proposal for the new list of outcomes.

These were also the outcomes shared with the students before the first test, and these were the list of outcomes I assessed this Spring.

To clarify, this selection and translation was done, not to circumvent the CQI process, but to make it possible to collect, for the first time, meaningful data, and thereby enable a well-informed discussion of the outcomes and curriculum redesign.

B2: Assessment Procedure

The assessment instruments were aspects of the assignments and problems from the tests. There were 19 assignments, mostly programming assignments, each with many subparts, all graded, but for outcomes assessment I chose only the most relevant. Note that almost all problems and most assignments included some element of challenge, such that to get 100% would require some extra insight or thought and the ability to explain very clearly. There were thus, unsurprisingly, no perfect test scores; indeed those who scored over 90% on the tests were true A+ stand-outs. Accordingly for most of the assessment instruments, adequate performance was not a perfect score, but attaining some fraction of the possible points. Below the evidence is summarized in the format x/y zz%, meaning that criterion on the problem was x points out of y possible, and that this criterion was met by zz% of the students. The percentage of students meeting criterion was estimated by sampling 10 students for each problem.

As all outcomes refer to "upon successful completion of this course, students will be able to ...", the performance of students with a D or lower grade should be excluded from the denominator. Only 87% of the students taking the final were thus relevant, so the criterion was accordingly adjusted from the standard of 70% demonstrating performance to 62%.

Overall, most of the outcomes were met (↑) or nearly met (↓), but several were far below that (⇓).

There seems to be a tendency for better performance on outcomes measured on assigments over those measured with test problems. The first topic covered, processes, appears to have been learned best. Some topics were clearly not satisfactorily learned, and these are addressed in the recommendations above.

B3: Assessment Results

Level 1

V1i. ↓ Choose a scheduling approach suitable for given simple problem.

> Test 1 "priority scheduling" 1/1 points 30%, Test 2.1 "thread preference" 2/3 40%, final "a scheduler is" 3/6 60%

V1j. ⇓ Explain segmentation and its security implications.

Test 1 "memory encryption" question 2/3 points 30%, final "every process" 1/1 40%

V1l. ↑ Explain some ways in which virtualization creates vulnerabilities.

Final "microcontrollers" question, 1/5 (since a level-2 question), 80%

V1m. ↑ Explain the components of process and virtual machine context.

Test 1 "to treat a process correctly" 9/13 30%, final "threads and processes" 2/4 50%, final "the kernel is responsible" 1/2 90%, final "os.environ" 1/3 80%

V1n. ⇓ Explain the need for paging, and basic strategies and their possible performance implications.

Final "paging" question 1/2, 40%

V1o. ↑ Describe the motivation for and gross characteristics of a trusted computing base

Test 2.1 "buggy" 2/3 80% "cloud" 1/1 30%

C1c. ↑ Given an application, identify the factors relevant to choosing a synchronous or asynchronous solution.

Test 2.1 "radiation" (a level-3 questions) 1/3 80%

E1f. ⇓ Choose when to use datagram versus virtual-circuit communication.

final "datagram" 3/6 30% not met

E1h. ↓ Define and compute simple transmission and propagation latencies.

Latency and Datarate Exercise 5/6 20%, final "correct equation" (1/1) 50%, "300KB message" 2/2 20%, "other factors" 2/2 70%

E1i. ↑ Explain how data is serialized (byte order, representation, buffering).

Sockets Asst 2 6/8 70%, Test 2.1 "recv(4096)" 2/4 50%

E1j. ↓ Explain the difference between lossy and lossless compression.

final "lossy... before" 1/1 60%

E1l. ↑ Interpret the output of a packet capture tool.

Packet Inspection Assignment 6/7 90%, Packet Parsing Assignment 4/6 70%

E1n. ↓ Explain the role of cryptographic hashes and symmetric and assymetric keys in security.

final "main technology" 2/3, 60%

E1o. ↓ Explain the basic concepts of DNS and IP.

final "think tank" 1/3 (since a level-3 question), 60%

E1p. ⇓ Explain the functionality handled at different network layers.

final "4-layer" 4.5/7 40%

E1q. ⇓ Explain some concepts in storing files on disk.

final "generally adjacent" 1/1 20% (question also involved networking)

E1r. ↑ Explain the memory hierarchy and the basic concepts of distributed storage.

final "order the following", 2/4 80%

E1s. ↓ Explain generic device APIs, including the bidirectional handling of interrupts and requests.

final "device driver", 2.5/4 60%

Level 2

V3x. ↓ Explain how domain names, IP addresses, file names, and memory segments are handled, and be able to identify efficiency and vulnerability implications.

Test 1 "virtual address" 2/4 10%, final "hierarchical naming", 2.5/4 70%, final "utep.bt" 2.5/4 50%

V2p. ↑ Explain how virtual machines, processes, containers and sandboxes work, and the implications for programmers using each.

Test 1 "traps", 3/3 90% "keep track of memory" 2/2 20%, Test 2.1 "buggy" 2/3 80%, final "virtualize servers" 1/2 50%

V2q. ↑ Explain process states and state transitions.

> Exercise 1 (Process States) 10/12 80%, Test 1 "Task Manager" 2/2 60%, "short-term scheduler" 1/1 100%

V2r. ↑ Explain the distinction between supervisor and user permissions, and what that implies for system calls, process management and networking; also explain the role of traps, especially in memory translation.

> Test 1 "traps", 3/3 90%

V2t. ↑ Write programs that use interprocess communication, namely pipes and sockets.

> Shell Assignment 20/30 60%, Sockets Asst 1 4/5 100%, Sockets Asst 2, 6/8 70%, Sockets Asst 3, 4/5 80%, Sockets Asst 4 4/5 80%

V2u. ↑ Use simple system calls for common needs.

> Shell Assignment 20/30 60%, Test 1 "fork 1" 1/1 30% "close(1), open()" 2/3 40%, "execv" Test 2.1 "locks t/f questions" 2/4 50%, Test 2.1 "synchronized withdrawal attacks" 2/4 80%

C2g. ↑ Build both ends of a simple producer-consumer link

> Producer Consumer Intro Asst 8/10 70%, Concurrency Beyond Locks Asst 6/8 80%

C2h. ↑ Build a server-side program that uses multi-threading to handle multiple simultaneous clients.

> Sockets Asst 5, 3/4 70%, Sockets Asst 6 3/4 70%

C2i. ↑ Identify situations where deadlock may occur, and suggest ways to prevent it.

> Deadlock Asst 3/3 100%, Test 2.1 "deadlock" 2/2 70%, Test 2.2 "deadlock" 2/4 90%

A2g. ⇓ Perform simple arithmetic computations related to major families (e.g. determine page number or whether an address is within a power-of-2 segment)

> Test 1 "virtual address" 2/4 10%; final "netmasks" 3/3 40%, final "router table" 4/4 40%

Level 3

V3w. ↓ When a process or a computer is running too slowly, infer some probable causes, including scheduling policy.

> final "a scheduler" 4/6 60%

C3j. ↓ Distinguish when blocking vs nonblocking calls are appropriate.

> final "nonblocking" 4/6, 50% (they would likely have done better with a more concrete question)

C3k. ↑ Correctly use counting semaphores etc. for queues handling for simple problems. (Note: after teaching the class, I would advocate changing this to "condition variables or semaphores")

> Test 2.1 "describe semaphore" 2/2 90%

Appendix C: 2018 Outcomes List with Commentary

Level 3: Analysis and synthesis

Virtualization:

V3v. Distinguish between policies and mechanisms in LDE context, Not assessed. One aspect of this is distinguishing specification from implementation, which has been tested in other courses, and needs to be merely reinforced. Other aspects are too advanced for this course. See also V2r.

V3w. Select (or diagnose) gross characteristics of relationship between scheduling policy and behavior,* Assessable restatement: When a process or a computer is running too slowly, be able to infer some probable causes.

V3x. Can analyze/select gross aspects of relationship between addressing family and application context,* Assessable restatement (Level 2) : Be able to explain how domain names, IP addresses, file names, and memory segments are handled, and be able to identify efficiency and vulnerability implications.

V3y. Analyze whether a function would be better implemented as a system or library call, Redundant to V1k, and assessed at Level 1, though arguably should be level 2.

V3z Security implications of various sandboxing strategies. This is an advanced topic, and was proposed for deletion in the Fall 2019 Systems Committee Report. Not assessed.

Concurrency:

C3j. Distinguish when blocking vs nonblocking calls are appropriate,* Assessed as stated.

C3k. Utilizing appropriate data structures for achieving synchronicity in a given problem* Assessable restatement (Level 2): Be able to correctly use counting semaphores, and queues for a simple problem.

Addressing:

A3h. Can select/analyze suitability of some addressing scheme for a simple problem Redundant to V3x.

Level 2: Application

Virtualization:

V2p. Choose appropriate container family (e.g. OS v. hw virtualization sandbox),* Assessable restatement: Explain how virtual machines, processes, containers and sandboxes work, and the implications for programmers using each.

V2q. Determine and motivate mapping between common process activities and canonical state,* Assessable Restatement: Explain process state and process scheduling. See also V3w.

V2r. Can characterize policies for common LDE contexts,* Assessable restatement: 1) Explain the distinction between supervisor and user permissions, and what that implies for system calls, process management and networking. 2) Explain the role of traps, especially in memory translation.

V2s. Characterize semantics and identify algorithms/data structures suitable for various memory allocation strategies, Not assessed. This is an advanced topic.

V2t. Can create and communicate among virtualized execution containers,* Assessable restatement: Be able to write programs that use interprocess communication, namely pipes and sockets.

V2u. Can construct programs using system and library interfaces.* I note that the use of libraries is learned in many other courses, the Fall 2019 Systems Committee Report recommended taking that out of this outcome. Assessed as: be able to use simple system calls for common needs.

Concurrency:

C2f. Use standard coordination primitives such as mutex to ensure a correctness property of a threaded program, Redundant to C3k.

C2g. Producer-consumer,* Assessable restatement: Build both ends of a simple producer-consumer link. See also C3k.

C2h. Construct a program that correctly responds to messages from multiple communication sources,* Assessable restatement: Build a server-side program that uses multi-threading to handle multiple simultaneous clients.

C2i. Deadlock, conditions under which it can occur, and standard approaches to avoid it.* Assessable restatement: Be able to detect situations where deadlock may occur, and to suggest ways to prevent it.

Addressing:

A2f. Can motivate the common uses and challenges of multiple of addressing schemes in the context of storage, memory allocation, and network addressing, Redundant to A3h and V3x.

A2g. Can perform simple arithmetic computations related to major families (e.g. determine page number or whether an address is within a power-of-2 segment) Assessed as stated. See also V3x.

Level 1: Familiarity

Virtualization:

V1f: Canonical process state diagram, Redundant to V2q.

V1g Difference between mechanism and policy, Redundant to V3v.

V1h. Describe and motivate role of traps, supervisor mode, and memory translation in implementing aspects of LDE (limited direct execution) Redundant to V2r.

*V1i, * Describe and motivate core principles of scheduling families. Eg. monoprogramming, prioritized, fairness, fifo, (non) preemptive, quantum,* Assessable restatement: Choose a scheduling approach suitable for given simple problem.

*V1j * Motivations for and gross characteristics of paged & segmented memory allocation strategies,* Assessable restatement: Explain segmentation and its security implications.

V1k. Differentiate between system/vm and library calls, Redundant to V3y.

*V1l * Gross security characteristics of virtualization,* Explain some ways in which virtualization creates vulnerabilities.

V1m. Components of process/vm context,* Assessed as stated. See V2p.

*V1n * Meaning and relevance of locality, LRU, NRU; relate them to demand loading/eviction as strategy to permit larger virtual memory size, .* Assessable restatement: Explain the need for paging, and basic strategies and their possible performance implications.

*V1o * Motivation for and gross characterization of trusted computing base* * Assessed as stated. Relates to V1k, V1l, etc.

Concurrency:

*C1c. * How asynchrony can simplify design, improve reliability and/or provide speedup,* Assessable restatement: Given a application, identify the factors relevant to choosing a synchronous or asynchronous solution.

C1d. Definitions and conditions related to incorrect behaviors under concurrency including (but not limited to) deadlock and inconsistency, Redundant to C2i.

C1e Principal of serialization bottlenecks as limiting factor in achieving speedup from concurrency. (generalized Amdahl's law). Assessed together with C1c.

Addressing:

A1c. Principles underlying and gross characteristics of hierarchical, uniform, and nonuniform address spaces, Assessed with V3x.

A1d. P2P architectures (flood, structured), Not assessed. An advanced topic.

A1e. Motivations for, principles, underlying structure, and scalability implications of segmented & paged memory; domain names, ports, IP and MAC. Not assessed. Redundant.

Encoding, persistence, and communication:

E1f. Motivations and gross characteristics of stream and datagram transport models,* Assessable restatement: Choose when to use datagram versus virtual-circuit communication.

E1g. Forwarding, Assessed with E1p.

E1h. Define and compute simple transmission and propagation latencies,* Relates to E1r. Assessed as stated.

*E1i * How data is serialized (byte order, representation, marshalling),* Assessed as stated.

E1j. Lossy v. lossless compression,* Assessable restatement: explain the difference between lossy and lossless compression.

E1k. Sharding with regards to distributed data stores, This is an advanced topic, and was proposed for deletion in the Fall 2019 Systems Committee Report. See E1r.

*E1l. * Inspection tools (fs editor & packet capture),* File system editing is an advanced topic. Assessable restatement: Be able to interpret the output of a packet capture tool.

E1m. Core ideas of ARQ, flow, and rate control, Not assessed. This is an advanced topic.

E1n. Gross characteristics of (a)symmetric crypto including key exchange; protocol security properties, This may cover cryptographic hashes also. Assessable restatement: explain how cryptography supports security.

*E1o. * Motivations and scalability of inter-address translation strategies (e.g. ARP, DNS. IP masquerading, subnet-driven IP routing),* Assessable restatement: Explain the basic concepts of DNS and IP.

E1p. Layering and end-to-end principles, as applied to eth and IP,* Explain the functionality handled at different network layers.

E1q. Data structures and their semantic/performance implications for inode, fat, iso filesystems,* This is an advanced topic, but a useful illustration of various methods and concepts, including fragmentation. Assessable subset: Explain some aspects of ways store files on disk.

E1r. * Timing and reliability characteristics of common random access mass storage devices; how redundancy can be used to increase reliability and performance,* Explain the memory hierarchy. Explain the basic concepts of distributed storage.

*E1s. * * Motivations for key aspects device driver design including (1) generic interfaces and (2) kernel/interrupt "halves".* The Fall 2019 Systems Committee Report recommended that this be renumbered to make it clear that this is related to virtualization. Assessable restatement: Explain generic device APIs. Explain the bidirectional handling of interrupts and requests.

thus reducing the list from 50-ish to 30-ish outcomes

For reference, here are the older Learning Objectives, from the 2015 syllabus.

> On successful completion of this course, students will

1. be able to apply the following in new situations:
a. operating system objectives and functions
b. process definition/description and control/management
c. threads, symmetric multiprocessing, microkernels
d. mutual exclusion and synchronization (software and hardware approaches): semaphores, monitors, message passing, readers/writers problem
e. concurrency: deadlock and starvation: principles of deadlock, deadlock prevention, avoidance, and detection
f. dining philosophers problem
g. memory management: paging, segmentation
h. virtual memory: hardware and control structures
i. scheduling algorithms

2. be able to apply:
a. file management (file organization, directories, and sharing), record blocking, secondary storage management
b. multiprocessor and real-time scheduling
c. I/O management and disk scheduling

3. have been introduced to:
a. Current windows operating system
b. UNIX operating system
c. distributed processing, client/server, and clusters
d. distributed process management

CS 4375 Proposed Changes

Systems Area CQI Committee (Moore, Freudenthal, Tosh, Ward), August 17

revised September 28 to include Proposed Change 4

Change 1: Rename as "Operating Systems Concepts"

Rationale: This is not a course in theory, so the current name, "Theory of Operating Systems" is inappropriate. We considered many alternatives, including "Systems 1" and "Operating Systems and Network Concepts," but agree that the title above is the most appropriate.

Change 2: Update the course description

Proposed: "Process and thread management, processor scheduling and concurrency, interprocess communication, memory management, input/output management, file systems, and networking basics"

The current description is "Process and thread management, concurrency, memory management, processor scheduling, I/O management and disk scheduling, and file management", but clearly this should include "interprocess communication" and "networking". "File management" should be simply "file systems". "Disk scheduling" should be omitted, since while it provides interesting case studies in algorithms and data structures, introduces no new concepts and it is seldom important for most purposes. Networking must be mentioned. "Security implications" is a recurring theme, but there is no specific module on this, and there are no assignments on this, so it probably should not be in the course description.

Change 3: Update the course outcomes

Proposed new outcomes:

Level 1

V1i. Choose a scheduling approach suitable for given simple problem.

V1j. Explain segmentation and its security implications.

V1l. Explain some ways in which virtualization creates vulnerabilities.

V1m. Explain the components of process and virtual machine context.

V1n. Explain the need for paging and the basics of demand loading.

V1o. Describe the motivation for and gross characteristics of a trusted computing base.

V1x. Explain how domain names, IP addresses, file names, and memory segments are handled.

C1c. Given an application, identify the factors relevant to choosing a synchronous or asynchronous solution.

E1f. Choose when to use datagram versus virtual-circuit communication.

E1h. Differentiate transmission and propagation latencies and some factors affecting them.

E1i. Explain how data is serialized (byte order, representation, buffering).

E1l. Interpret the output of a packet capture tool.

E1n. Explain the role of cryptographic hashes and symmetric and assymetric keys in security.

E1o. Explain the basic concepts of DNS and IP. (Eric promises to suggest a rewording)

E1p. Explain the functionality handled at different network layers.

E1q. Explain some concepts in storing files on disk.

E1r. Explain the memory hierarchy and the basic concepts of distributed storage.

E1s. Explain generic device APIs, including the bidirectional handling of interrupts and requests.

Level 2

V2q. Use the concepts of process state and state transition to characterize system and process behavior.

V2r. Relate the distinction between supervisor and user permissions to the design and implementation of system calls.

V2t. Write programs that use interprocess communication, specifically pipes and/or sockets.

V2u. Use simple system calls for common needs.

C2g. Implement producer-consumer coordination.

C2h. Build a server-side program that uses multi-threading to handle multiple simultaneous clients.

C2i. Identify situations where deadlock may occur, and suggest ways to prevent it.

A2g. Perform simple arithmetic computations related to major families (for example determine page number or whether an address is within a power-of-2 segment)

Level 3

V3w. When a process or a computer is running too slowly, infer some probable causes.

V3p. Choose among virtual machines, processes, containers and sandboxes as ways to support common programmer needs.

C3j. Distinguish when blocking versus nonblocking calls are appropriate.

(Note: For discussion purposes the numbering above reflects the 2018 outcomes. After approval they should be renumbered.)

Background

The 2018 outcomes were judged problematic in that:

- there are too many to teach
- some are aspirational mentions of advanced topics that cannot be covered in a first systems course.
- there are too many to assess (50), including many which are redundantly stated at different levels
- many are not directly assessable, being instead just mentions of important concepts, or of technologies that illustrate those concepts, or of interesting juxtapositions or abstractions that relate to deeper understandings of those concepts
- many are worded in ways that do not enable students to understand what they mean or what is expected.

Recommendations addressing some of these issues were made in the Fall 2019 Systems Committee CQI Report, but apparently never acted upon.

In December Freudenthal helped Ward understand the intent behind the current outcomes. Ward came to understand that, behind the details, there are a few overarching learning goals for the course, which he summarized as:

> Learn concepts that will be foundational for further study, whether course-based or on-the-job, of:
>
> 1. Computer Security and Computer Forensics
> 2. Systems Administration and Network Administration
> 3. Systems Programming and Network Programming, including developing for non-standard platforms, such as embedded systems, systems in the cloud, and high-performance (parallel) systems
> 4. Effective use of modern computers, notably Windows and Posix systems

Ward then worked to identify the essential learning outcome (marked * in Appendix 1) and restate each in a way both assessable and comprehensible to students, and then refined these in discussion with Freudenthal.

Ward used that preliminary version of these outcomes in pilot form for Section 2 of OS in Spring 2021, and overall they seemed comprehensible to the students, possible to teach, possible to learn (given the constraints, for most students), and possible to assess.

The committee then refined the outcomes based on experience in that class, as documented in its CQI report, and other considerations.

Below is a point-by-point explanation of how the proposed outcomes compare to and improve on the current outcomes.

Change 4: Renumber to be 3000 level

This is proposed to encourage students to take this course earlier than most electives.

Appendix 1: 2018 Outcomes List with Commentary

Level 3: Analysis and synthesis

Virtualization:

V3v. Distinguish between policies and mechanisms in LDE context, Not assessed. One aspect of this is distinguishing specification from implementation, which has been tested in other courses, and needs to be merely reinforced. Other aspects are too advanced for this course. See also V2r.

V3w. Select (or diagnose) gross characteristics of relationship between scheduling policy and behavior,* Assessable restatement: When a process or a computer is running too slowly, be able to infer some probable causes.

V3x. Can analyze/select gross aspects of relationship between addressing family and application context,* Assessable restatement (Level 2) : Be able to explain how domain names, IP addresses, file names, and memory segments are handled, and be able to identify efficiency and vulnerability implications.

V3y. Analyze whether a function would be better implemented as a system or library call, Redundant to V1k, and assessed at Level 1, though arguably should be level 2.

V3z Security implications of various sandboxing strategies. This is an advanced topic, and was proposed for deletion in the Fall 2019 Systems Committee Report. Not assessed.

Concurrency:

C3j. Distinguish when blocking vs nonblocking calls are appropriate,* Assessed as stated.

C3k. Utilizing appropriate data structures for achieving synchronicity in a given problem* Assessable restatement (Level 2): Be able to correctly use counting semaphores, and queues for a simple problem.

Addressing:

A3h. Can select/analyze suitability of some addressing scheme for a simple problem Redundant to V3x.

Level 2: Application

Virtualization:

V2p. Choose appropriate container family (e.g. OS v. hw virtualization sandbox),* Assessable restatement: Explain how virtual machines, processes, containers and sandboxes work, and the implications for programmers using each.

V2q. Determine and motivate mapping between common process activities and canonical state,* Assessable Restatement: Explain process state and process scheduling. See also

V3w.

V2r. Can characterize policies for common LDE contexts,* Assessable restatement: 1) Explain the distinction between supervisor and user permissions, and what that implies for system calls, process management and networking. 2) Explain the role of traps, especially in memory translation.

V2s. Characterize semantics and identify algorithms/data structures suitable for various memory allocation strategies, Not assessed. This is an advanced topic.

V2t. Can create and communicate among virtualized execution containers,* Assessable restatement: Be able to write programs that use interprocess communication, namely pipes and sockets.

V2u. Can construct programs using system and library interfaces.* I note that the use of libraries is learned in many other courses, the Fall 2019 Systems Committee Report recommended taking that out of this outcome. Assessed as: be able to use simple system calls for common needs.

Concurrency:

C2f. Use standard coordination primitives such as mutex to ensure a correctness property of a threaded program, Redundant to C3k.

C2g. Producer-consumer,* Assessable restatement: Build both ends of a simple producer-consumer link. See also C3k.

C2h. Construct a program that correctly responds to messages from multiple communication sources,* Assessable restatement: Build a server-side program that uses multi-threading to handle multiple simultaneous clients.

C2i. Deadlock, conditions under which it can occur, and standard approaches to avoid it.* Assessable restatement: Be able to detect situations where deadlock may occur, and to suggest ways to prevent it.

Addressing:

A2f. Can motivate the common uses and challenges of multiple of addressing schemes in the context of storage, memory allocation, and network addressing, Redundant to A3h and V3x.

A2g. Can perform simple arithmetic computations related to major families (e.g. determine page number or whether an address is within a power-of-2 segment) Assessed as stated. See also V3x.

Level 1: Familiarity

Virtualization:

V1f: Canonical process state diagram, Redundant to V2q.

V1g Difference between mechanism and policy, Redundant to V3v.

V1h. Describe and motivate role of traps, supervisor mode, and memory translation in implementing aspects of LDE (limited direct execution) Redundant to V2r.

*V1i, * Describe and motivate core principles of scheduling families. Eg. monoprogramming, prioritized, fairness, fifo, (non) preemptive, quantum,* Assessable restatement: Choose a scheduling approach suitable for given simple problem.

*V1j * Motivations for and gross characteristics of paged & segmented memory allocation strategies,* Assessable restatement: Explain segmentation and its security implications.

V1k. Differentiate between system/vm and library calls, Redundant to V3y.

*V1l * Gross security characteristics of virtualization,* Explain some ways in which virtualization creates vulnerabilities.

V1m. Components of process/vm context,* Assessed as stated. See V2p.

*V1n * Meaning and relevance of locality, LRU, NRU; relate them to demand loading/eviction as strategy to permit larger virtual memory size, .* Assessable restatement: Explain the need for paging, and basic strategies and their possible performance implications.

*V1o * Motivation for and gross characterization of trusted computing base* * Assessed as stated. Relates to V1k, V1l, etc.

Concurrency:

*C1c. * How asynchrony can simplify design, improve reliability and/or provide speedup,* Assessable restatement: Given a application, identify the factors relevant to choosing a synchronous or asynchronous solution.

C1d. Definitions and conditions related to incorrect behaviors under concurrency including (but not limited to) deadlock and inconsistency, Redundant to C2i.

C1e Principal of serialization bottlenecks as limiting factor in achieving speedup from concurrency. (generalized Amdahl's law). Assessed together with C1c.

Addressing:

A1c. Principles underlying and gross characteristics of hierarchical, uniform, and nonuniform address spaces, Assessed with V3x.

A1d. P2P architectures (flood, structured), Not assessed. An advanced topic.

A1e. Motivations for, principles, underlying structure, and scalability implications of segmented & paged memory; domain names, ports, IP and MAC. Not assessed. Redundant.

Encoding, persistence, and communication:

E1f. Motivations and gross characteristics of stream and datagram transport models,* Assessable restatement: Choose when to use datagram versus virtual-circuit communication.

E1g. Forwarding, Assessed with E1p.

E1h. Define and compute simple transmission and propagation latencies,* Relates to E1r. Assessed as stated.

*E1i * How data is serialized (byte order, representation, marshalling),* Assessed as stated.

E1j. Lossy v. lossless compression,* Assessable restatement: explain the difference between lossy and lossless compression. However this was apparently inherited from some security curriculum requirement which no longer applies anyway.

E1k. Sharding with regards to distributed data stores, This is an advanced topic, and was proposed for deletion in the Fall 2019 Systems Committee Report. See E1r.

*E1l. * Inspection tools (fs editor & packet capture),* File system editing is an advanced topic. Assessable restatement: Be able to interpret the output of a packet capture tool.

E1m. Core ideas of ARQ, flow, and rate control, Not assessed. This is an advanced topic.

E1n. Gross characteristics of (a)symmetric crypto including key exchange; protocol security properties, This may cover cryptographic hashes also. Assessable restatement: explain how cryptography supports security.

*E1o. * Motivations and scalability of inter-address translation strategies (e.g. ARP, DNS. IP masquerading, subnet-driven IP routing),* Assessable restatement: Explain the basic concepts of DNS and IP.

E1p. Layering and end-to-end principles, as applied to eth and IP,* Explain the functionality handled at different network layers.

E1q. Data structures and their semantic/performance implications for inode, fat, iso filesystems,* This is an advanced topic, but a useful illustration of various methods and concepts, including fragmentation. Assessable subset: Explain some aspects of ways store files on disk.

E1r. * Timing and reliability characteristics of common random access mass storage devices; how redundancy can be used to increase reliability and performance,* Explain the memory hierarchy. Explain the basic concepts of distributed storage.

*E1s. * * Motivations for key aspects device driver design including (1) generic interfaces and (2) kernel/interrupt "halves".* The Fall 2019 Systems Committee Report recommended that this be renumbered to make it clear that this is related to virtualization. Assessable restatement: Explain generic device APIs. Explain the bidirectional handling of interrupts and requests.

Appendix 2: 2015 List

For reference, here are the older Learning Objectives, from the 2015 syllabus.

> On successful completion of this course, students will
>
> 1. be able to apply the following in new situations:
> a. operating system objectives and functions
> b. process definition/description and control/management
> c. threads, symmetric multiprocessing, microkernels
> d. mutual exclusion and synchronization (software and hardware approaches): semaphores, monitors, message passing, readers/writers problem
> e. concurrency: deadlock and starvation: principles of deadlock, deadlock prevention, avoidance, and detection

f. dining philosophers problem
g. memory management: paging, segmentation
h. virtual memory: hardware and control structures
i. scheduling algorithms

2. be able to apply:
a. file management (file organization, directories, and sharing), record blocking, secondary storage management
b. multiprocessor and real-time scheduling
c. I/O management and disk scheduling

3. have been introduced to:
a. Current windows operating system
b. UNIX operating system
c. distributed processing, client/server, and clusters
d. distributed process management