1. About this Course

Conceived and developed at Boston University in the mid 1990s, this course is *unique* in its treatment of computing systems by recognizing that there is a common denominator of concepts that lies at the core of the various types of computing systems that one may encounter or develop. Broadly speaking, the "common denominator of computing systems concepts" covered in this course revolve around three key topics: (1) resource management, (2) concurrency management, and (3) performance modeling and evaluation.

Rather than looking at specifics of particular instances of computing system, each of which is typically covered in a separate course (e.g., operating systems, networking systems, embedded systems, distributed systems, database systems, cloud computing systems, Internet of Things, cyber-physical systems, etc.), this course looks at what is common across all these systems. The advantage of this approach is that if focuses the course on concepts that are mostly independent of technology, and thus *timeless* – hence its name "Fundamentals of Computing Systems". Indeed, the main motive for developing this course back in the mid-1990s is the realization that computing systems come (and will continue to come) in many flavors, which underscored the need to develop a single course that covered the key aspects related to all these variants, which we believe any computer science graduate should master.

The philosophy underlying the design of this course is that students should be familiarized with problems that reoccur in software systems, and should be acquainted with the set of classical algorithms and techniques for solving these problems, independent of the context in which these problems arise. In particular, it is important to develop the ability to recognize standard problems in different wordings and within unusual context, and match them with appropriate solutions.

To ensure that students do not mix these key fundamental concepts with technology-specific or application-specific details, in this course what constitutes a "computing system" is intentionally generic. This allows the course to achieve its objective of focusing on the basic reasoning approaches,

mathematical tools, and modeling techniques that can be applied to a multitude of system instances. By the end of the course, students who take CS-350 will develop the ability to map new problems onto existing solutions, or at the very least onto existing approaches towards a solution. In fact, this course will help the careful student realize that many system instances have a common denominator of challenges and issues that can be approached using well established techniques and abstractions.

It is very important to understand that this course is NOT an operating systems course – or for that matter a computer architecture course or a networking systems course or a distributed systems course, etc. It is a course on (most) of the fundamental concepts that underlie the design and implementation of *all* of these systems, including that of an operating system (which, by virtue of its function, is the one computing system through which most of the concepts covered in this course could be treated. In particular, issues related to resource management, concurrency management and synchronization are central to an operating system.

The reason this point cannot be over-emphasized enough is because many developers overlook the same issues that operating systems must deal with when building software artifacts. The results are (not surprisingly) either buggy or inefficient, and seldom salvageable in the sense that the right approach to fix these bugs and inefficiencies is typically to rewrite the whole system!

To some extent, the understanding and mastery of issues related to (1) resource management, (2) concurrency management, and (3) performance modeling and evaluation is really what differentiate software development by a Computer Scientist from that done by a certified hacker. The course covers these issues through an examination of the following topics:

1. Abstraction of systems into fundamental sub-components;
2. Metrics and trade-offs of system performance;
3. Modeling and analysis of systems as networks of queues;
4. Design and evaluation of resource management policies;
5. Synchronization and concurrency management;
6. Scalability through parallelism and distribution.

While there are dozens of great textbooks by major publishers (costing north of a hundred dollars each) that cover specific types of systems (operating systems, distributed systems, networking systems, database systems), there is no single textbook that adopts this course's philosophy. This made it necessary for the course to rely on a set of lecture notes that have been developed and refined over the years, culminating in the current work-in-progress textbook that represents the integration of materials from these notes. Better yet, it is free!

Enjoy the ride!