CSCE 990: Real-Time Systems
Course Syllabus
Fall 2007

Instructor:	Prof. Steve Goddard	**Office:**	Avery 357
E-mail:	goddard@cse.unl.edu	**Telephone:**	472-9968
Office Hours:	12:00-1:30pm Tu, 12:00-1:30pm Th or by appointment		

Meeting Place: Avery 118
Meeting Time: 2:00 - 3:15pm, TuTh
Course Web Page: http://www.cse.unl.edu/∼goddard/Courses/RealTimeSystems

Text: *Real-time Systems*, Jane Liu, Prentice Hall, 2000.

Prerequitsites: CSCE 451/851: Operating Systems Principles.

Grading:

Homework & Programming	20%
Project	30%
Midterm Exam	20%
Final Exam	30%

We will probably have three homework assignments. Most of these will involve programming. All homework submitted after its deadline is considered late. Assignments that are submitted within 24 hours after the original deadline are considered to be "one day late," within 48 hours, "two days late," etc. A late homework assignment will be accepted without penalty if the following conditions are met:

- the total "lateness" of all homework assignments received to date (including the current assignment) does not exceed 3 days.
- the student does not miss class on the day the assignment is due. Exceptions to this requirement must be approved by the instructor in advance.

Late assignments must be hand delivered to either the instructor or the TA, or emailed to the instructor. The penalty for late assignments is 25% per day they are late. An assignment that is 4 days late will receive no credit. Weekends count in evaluating the lateness of an assignment.

Each student must complete a class project. You are responsible for defining your own project. Your project can be either an experimental investigation or a survey or research paper. The project must be a fairly significant piece of work — while I'm not necessarily looking for something publishable (although that would be great), a 10-page survey paper is not going to cut it. The project will be due Wednesday, December 12, which is the last day an assignment can be due during dead week (and

the semester.) Consider this your notification of an assignment due during dead week. There are no "free late days" for Projects. You will loose 25% per day your project is late.

There will be one midterm exam. The final exam for this course is scheduled for Tuesday, December 18 at 1:00 pm. The final exam covers the entire course.

Important Note: Students will also be graded on class participation. I reserve the right to adjust final grades by up to half a letter grade (positive or negative) as a "reward" for in-class participation.

Course Conduct and Academic Integrity
Students are *encouraged* to work together on homework assignments. *Acceptable collaboration* on homework includes:

- discussing the assigned problems to understand their meaning,
- discussing possible approaches to assigned problems,
- discussing the UNIX system features, or general programming principles in the solution of programming problems related to the assignments.

In all cases you must explicitly acknowledge any and all substantive help received from other individuals during the course of the preparation of your homework solution. That is, if you collaborate with other individuals then you must include an explicit acknowledgment in your homework solution of the persons from whom you received aid.

Unacceptable collaboration, unless explicitly stated, on homework includes:

- copying (verbatim use) of physical papers or computer files.[1]
- submission of solutions that are jointly authored, or authored either wholly or in part by other individuals (unless the assignment is a group project).

The general rule to be followed is that the strategy and approach of solutions may be developed jointly but *all* actual solutions (i.e., the final solution) must be constructed and written up individually. Work done jointly should not be done in sufficient detail as to make it a solution. For example, the design of a program solution may be performed jointly, however, each student must write all the code they eventually submit as their solution. *No code may be shared between students, unless the assignment is a group project*. Similarly, for written assignments, solutions may be *sketched* out jointly, however each student must construct the final form of their solution individually and write-up their own solution. You will be held accountable if someone else copies your work, even if you are unaware of the event. Thus, you should make sure all of your files are properly secured.

Unacceptable collaboration will be considered a violation of the Student Code of Conduct, and will result in a failing grade for the course. In other words: if you cheat, you will fail! In addition, the incident will be reported to the CSE Department, in accordance with new policy on academic

[1] This includes computer files that are copied and then edited and/or reformatted.

integrity. You are responsible to read the department policy and adhere to it.

Should questions arise the course of working on a problem please feel free to immediately contact the instructor either by telephone, electronic mail, or by an office visit. In principle, if you work with others in good faith and are honest and generous with your attributions of credit you will have no problems.

Special Needs
Students with disabilities are encouraged to contact Christy Horn for a confidential discussion of their individual needs for academic accommodation. It is the policy of the University of Nebraska-Lincoln to provide flexible and individualized accommodation to students with documented disabilities that may affect their ability to fully participate in course activities or to meet course requirements. To receive accommodation services, students must be registered with the Services for Students with Disabilities (SSD) office, 132 Canfield Administration, 472-3787 voice or TTY.

Topics: The rest of this syllabus is the list of topics I plan to cover. (Chapter numbers refer to Liu's book.) The last 5 weeks, topics starting with mixing real-time and non-real-time, are subject to change. We will, as a class, decide the focus of those weeks, if we do not run out of time. I am thinking it might be fun to explore new research results generated here at UNL, or have some of you present papers.

Part I: Uniprocessor Scheduling of Independent Tasks.

- **Introduction to real-time systems (1 week).**
  - J. Stankovic, "Misconceptions About Real-Time Computing," *IEEE Computer*, Vol. 21, No. 10, October 1988, pp. 10-19.
  - Chapter 1: Example real-time applications.
  - Chapter 2: Hard vs. soft real time.
  - Chapter 3: Reference model (includes lots of definitions used in later chapters).
- **Classic uniprocessor scheduling results (4 weeks).**
  - Static scheduling.
    - Chapter 5: Cyclic executives.
  - Dynamic scheduling.
    - Dynamic-priority scheduling:
      - Chapter 4, Section 6: Optimality of EDF and LLF.
      - Chapter 6, Section 3: Utilization-based schedulability test for EDF.
      - Nonpreemptive EDF from:
        K. Jeffay, D. Stanat, and C. Martel, "On Optimal, Non-Preemptive Scheduling of Periodic and Sporadic Tasks," *Proceedings of the 12th IEEE Real-Time Systems Symposium*, San Antonio, TX, December 1991, pp. 129-139.

    - Static-priority scheduling:
        - Chapter 6, Section 4: Optimality of RM and DM.
        - Chapter 6, Section 7: Utilization-based schedulability test for RM. (Skip 6.7.3 – 6.7.5.)
        - Chapter 6, Sections 5 and 6: Demand-based scheduling conditions for static-priority systems.
    - Dealing with complexities arising in real systems.
        - Chapter 6, Section 8: Practical considerations. (Skip 6.8.6 – 6.8.7.)
        - Timing analysis, from:
          C.M. Krishna and K.G. Shin, *Real-Time Systems*, McGraw Hill, pages 25-37.

- **Some real systems (or, how much of this theory really gets used anyway?) (1 week).**
  - Chapter 12, Sections 1 and 2: Basic operating-system functions needed for real-time computing.
  - Chapter 12, Sections 6 and 7: A brief survey of commercial real-time and non-real-time operating systems.

- **Intractability results (2 weeks).**
  - Preemptive systems.
    - Dynamic-priority systems, from:
      S. Baruah, R. Howell, and L. Rosier, "Feasibility Problems for Recurring Tasks on One Processor," *Theoretical Computer Science*, Vol. 118, pp. 3-20, 1993.
    - Static-priority systems, from:
      J. Leung and J. Whitehead, "On the Complexity of Fixed-Priority Scheduling of Periodic, Real-Time Tasks," *Performance Evaluation*, Vol. 2, No. 4, pp. 237-250, 1982.
  - Nonpreemptive systems.
    - Dynamic-priority systems, from:
      K. Jeffay, D. Stanat, and C. Martel, "On Optimal, Non-Preemptive Scheduling of Periodic and Sporadic Tasks," *Proceedings of the 12th IEEE Real-Time Systems Symposium*, San Antonio, TX, December 1991, pp. 129-139.
    - Static-priority systems (no good reference here).
  - A bullet-list survey of known results about uniprocessor scheduling.

    48 task models can be defined for uniprocessors based on the following criteria:
    - Preemptive versus nonpreemptive scheduling.
    - Dynamic versus static priorities.
    - Synchronous versus asynchronous job releases.
    - Deadlines equal periods, or deadlines are less than or equal to periods, or deadlines are arbitrary.
    - Periodic versus sporadic tasks.

For each model, we will indicate which scheduling algorithm (if any) is known to be optimal. We will also categorize the feasibility problem as either solvable in polynomial time, solvable in pseudo-polynomial time, co-NP-hard in the strong sense, or "don't know." You may be surprised to see how many "don't knows" there are. Many of the results summarized here will have been discussed in class by this point. Others can be found in the literature.

Part II: Beyond Uniprocessor Independent Task Models.

- **Resource sharing (2 weeks).**
  - **Motivation:** "What Really Happened on Mars Rover Pathfinder," by Mike Jones.
  - Chapter 8: Priority inheritance and priority ceiling protocols, stack resource protocol. (Skip 8.7 – 8.10.)
  - Resource sharing under EDF, from:
    K. Jeffay, "Scheduling Sporadic Tasks with Shared Resources in Hard-real-time Systems," *Proceedings of the 13th IEEE Real-time Systems Symposium*, Phoenix, AZ, December 1992, pp. 89-99.
  - Lock-free approach, from:
    J. Anderson, S. Ramamurthy, and K. Jeffay, "Real-Time Computing with Lock-Free Objects," *ACM Transactions on Computer Systems*, Vol. 15, No. 6, pp. 388-395, May 1997.
- **Mixing real-time and non-real-time (2 weeks).**
  - Chapter 7, Section 1: Introduction.
  - Chapter 7, Section 2: Deferrable servers.
  - Chapter 7, Section 3: Sporadic servers.
  - Chapter 7, Section 4: Constant utilization and total bandwidth servers. (We will skip weighted fair queuing, since we are covering proportional-share scheduling, which is similar.)
- **Fairness (1.5 week).**
  - Proportional-share scheduling, from:
    I. Stoica, H. Abdel-Wahab, K. Jeffay, S. Baruah, J. Gehrke, and C.G. Plaxton, "A Proportional Share Resource Allocation Algorithm for Real-Time, Time-Shared Systems," *Proceedings of the 17th IEEE Real-Time Systems Symposium*, pp. 288-299, 1996.
  - Pfair scheduling, from:
    S. Baruah, N. Cohen, C.G. Plaxton, and D. Varvel, "Proportionate Progress: A Notion of Fairness in Resource Allocation," *Algorithmica*, Vol. 15, pp. 600-625, 1996.
- **Multiprocessors and distributed systems (1.5 weeks).** (If we are running short of time, this will be the first topic to be cut.)
  - Chapter 9, Section 3: Multiprocessor priority ceiling protocol.
  - Chapter 9, Section 4: End-to-end scheduling.