OPERATING SYSTEMS

Ariel University
Faculty of Natural Sciences
Department of Computer Science

DR. ELAD AIGNER-HOREV

GENERAL INFORMATION

Academic credit points: 2.5

Prerequisites:

1. Introduction to Object Oriented Programming (or Java programming).
2. Data structures.
3. Introduction to computers and the C programming language.
4. Introduction to computation.
5. Computer Networks.
6. Algorithms 1.
7. Algorithmic Number Theory.
8. Discrete Math.
9. Linear Algebra 1& 2.

(*) If you encounter a different version of this syllabus with less prerequisites then the version with more prerequisites prevails and is the one we shall respect.

Degree: mandatory course for the B.Sc. program in Computer Science.

Schedule:

2 academic hours of lecture
1 academic hour of practical session

Method of instruction:

- Frontal lectures and practical session that will not take place in a computer lab.
- A large scale programming assignment implementing the paradigms seen in class.
- Small verification assignments
- Reading assignments
- Course booklet
- Slides for practical sessions.
- Instructed self-study.

COURSE DESCRIPTION AND GOALS

The operating system (OS, henceforth) is the software in charge of managing the resources of our computational device. There is a great variety of OSs; each dedicated to its own computational environment. In this course, the interest falls on the features that most commonly appear in OSs. After identifying the main components and basic design of common modern OSs, the lectures of the course will mainly focus on the topic of *synchronisation algorithms*. A detailed explanation of this topic can be found below. The lectures will not involve any specific programming languages and the discussion during the lectures will kept abstract. This is then enhanced by the practical sessions that will exemplify and will delve into much details regarding *Linux System Programming* and *Linux Shell Programming.*

* The course entails heavy programming in C and Java. Do not take this course if you are not in full command of any of these programming languages.
* Almost all lectures will involve proving correctness of synchronisation algorithms of various kinds. A certain level of mathematical maturity is expected (see prerequisite courses).
* There is a heavy dependence on data structures and their manipulation; in particular in this course we will have several processes/threads manipulating a shared data structure.
* Except for the frontal lectures and practical sessions which will be conducted in Hebrew, all other interaction with the students will be done in English.

COURSE PLAN

The following is a non-exhaustive overview regarding the topics that will be considered throughout the course. We make the following disclaimer:

1. The order and the extent of each topic is subject to change according to our discretion.
2. We reserve the right to enhance the list below as we see fit; introduce generalisations that we deem relevant even if those are not on the list.
3. We reserve the right to switch topics between lectures and practical session as we see fit.
4. We also reserve the right to designate topics for self-study as we see fit.
5. More concisely: the list below is not binding in any way shape or form.

Lectures: tentative non-binding plan

1. **Introduction.**
   I. Definition of core concepts in OSs: processes, threads, files, systems calls, shell, synchronous vs a synchronous systems, the booting process, interrupts.
   II. Process Scheduling.
   III. Virtual memory.
2. **The mutual exclusion problem.** Definition and properties of synchronisation algorithms.
3. **Early synchronisation patterns.**
   I. Lock variables.
   II. Strict alternation pattern.
4. **Synchronisation using atomic operations.**
   I. Test-and-set operation.
   II. The test-and-test-and-set operation
   III. The swap operation
   IV. The compare&swap operation
5. **Dekker's algorithm**

6. **Peterson's algorithm**
7. **Semaphores.**
    I. Definitions: classical semaphores, bounded semaphores, binary semaphores.
    II. The singling pattern.
    III. The mutex algorithm.
    IV. The rendezvous pattern.
    V. The multiplex pattern.
    VI. Barrier synchronisation.
    VII. Implementing counting semaphores using binary semaphores.
    VIII. The consumer-produce problem.
    IX. The readers-writers problem.
    X. The one way tunnel problem.
    XI. The dinning philosophers problem.
8. **Monitors**.
9. **The filter algorithm.**
10. **The tournament algorithm.**
11. **Lamport's bakery algorithm.**
12. **The black and white bakery algorithm.**
13. **Contention detection algorithms.**
14. **Aravind's algorithm.**
15. **Lamport's fast mutex algorithm.**
16. **Mutex-free and lock-free synchronisation algorithms.**
17. **Concurrent data structures.**

Practical sessions: tentative non-binding plan

1. Processes in Linux.
2. Signals in Linux.
3. The Linux file system.
4. Programming Pthreads.
5. Synchronisation constructs in Linux:
    1. Semaphores
    2. Locks
    3. Reentrant Locks
    4. Monitors
    5. Events
6. Memory management in Linux:
    I. Virtual memory.
    II. Paging algorithms.
7. Inter-process communication: Pipes, message queues, and shared memory.
8. Device drivers and modules in Linux.

Course Assignments

There are two types of assignments in this course.

1. **Large programming assignment.** The course involves **one non-mandatory** large scale programming assignment encompassing the entire semester to be done only in C over Linux (or C++ for the enthusiasts) .
   I. The assignment will require the students to engage heavily in **self-study** of various topics related to programming, data structures, algorithm design and analysis, databases, and networking.
   II. Below we explain how this non-mandatory assignment will be used for the calculation of the final grade.
   III. The assignment will be graded via a frontal examination.
   IV. The material covered by this assignment is part of the material for the final exams including all self-study material.

2. **Verification assignments.** We will also hand out smaller assignments throughout the course that are also not mandatory. The aim of these assignment is to help the students keep a steady focus on the course. Verification assignments include reading assignments as well.

While all assignments are not mandatory the working assumption is that the student body is relentlessly pursuing the assignments handed out. It is the students' choice whether to do the assignments or not and with what rigour. We strongly advise that one does these assignments with outmost care and our assumption is that this is the reality.

+ The verification assignments are by no means a substitute for the major programming assignment.
+ The major programming assignment is by no means a substitute for the verification assignments.
+ Doing the assignments of the course correctly constitutes the bare minimum effort required in this course; as such merely doing the assignments bares no promise as to your final grade or success in the final exams. For an above passing grade more effort must be put into the course.

Course lecture booklet

A booklet containing all lecture material will be handed out via the course website. Students bare full responsibility as to its content whether this has been taught in class or not.

Practical session slides

Each practical session will be accompanied with slides. Students bare full responsibility for the content of these slides whether these have been presented in class or not.

Course website

The course has a website address of which will be supplied to the student at the first lecture. Students are responsible for keeping track after the site and all its content. All information appearing in the site is part of the final exams.

Self-study

Self-study is a core method in this course and will be used quite heavily. Any material assigned for self-study is a part of the material for the final exam and will be treated as though it was taught in class.

1. All material found in the website that was not covered in class is designated as self-study for which students bare full responsibility.
2. The students are to learn on their own how to use the gcc compiler.
3. All self-study material is part of the material for the final exams.

Grade composition

Passing. In order to pass the course the final grade has to be ≥ 60.

Calculation of the final grade will be done as follows. Let *t* denote the grade of the final exam, let *a* denote the grade of the semester long programming assignment, and let *f* denote the final grade. The determination of the final grade depends on whether a student chooses to submit the course programming assignment or not.

1. If a student submits the course programming assignment then

$$f = 0.75*t + 0.25*a$$

2. otherwise; if a student chooses to not submit the course programming assignment then

$$f = t$$

Attendance in the lectures and the practical sessions is not mandatory. Nevertheless, students are fully responsible for keeping track with the course in case they choose not to attend the lectures and sessions.

The final exam.

1. The final exams will be conducted in a regular class with no access to any computational devices.
2. The exams will be handwritten.
3. No material of any kind is allowed in the final exams.
4. The duration of the final exam will be **2-3 hours** depending on the decision of the lecturer**.**
5. English will be used in the final exams.
6. All material covered by the course programming assignment is a part of the material for the final exams; this includes the self-study material included in the assignment.
7. All material covered by verification assignments is a part of the material for the final exams.
8. All material appearing in the course lecture booklet is a part of the material for the final exams.
9. All material appearing in the slides of the practical session is a part of the material for the final exams.
10. All material appearing in the course website is a part of the material for the final exams.
11. Additional information as to the structure of the final exam etc. will be provided according to the lecturer's discretion during the course.
12. Additional guidelines regarding the exams will be delivered during the semester and those would be binding.

Exams from previous years pose no restriction or binding obligations for this course. This applies to the structure of the final exams, content, and focus. Students are hereby formally discouraged from assuming anything regarding the final exams of this course based on exams of previous instances of this course. Put another way, exams from previous years are utterly meaningless as far as this course is concerned.

LITERATURE

1. A. S. Tanenbaum, *Modren Operating Systems,* 3rd edition.
2. A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating Systems Concepts,* 9th edition.
3. T. W. Doeppner, *Operating Systems in Depth.*
4. W. Stallings, *Operating Systems: internals and design principles*, *7th edition.*
5. *Course booklet*
6. *Course slides*