SAULT COLLEGE OF APPLIED ARTS & TECHNOLOGY
SAULT STE. MARIE, ONTARIO

COURSE OUTLINE

COURSE TITLE: **Technical Systems Programming I**

CODE NO.: **CSD305** SEMESTER: **5**

PROGRAM: COMPUTER ENGINEERING TECHNOLOGY

AUTHOR: Fred Carella

DATE: Sep 1998 PREVIOUSLY DATED: Sep 1997

APPROVED: L DeRusavit Aug 26/98
DEAN **DATE**

Length of Course: **16 weeks**
Prerequisites: **CSD205** Total Credit Hours: **96**

Technical Systems Programming I — CSD305

COURSE NAME — COURSE CODE

I. COURSE DESCRIPTION:

This course is intended to provide a firm foundation in the programming of various computer operating systems and to develop the students ability to write applications in modern programming environments. In particular, programs will be written under the UNIX and Microsoft Windows operating system environments. It introduces the student to writing programs within the Windows environment using the Visual C++ programming language. Object oriented programming techniques will be used with emphasis on programming with the Microsoft Foundation Classes (MFC). The UNIX environment will be programmed using C. The operating systems used will be Windows 95 and LINUX. This is the first of two courses in technical systems programming which will develop the students ability to program in various operating systems.

II. TOPICS TO BE COVERED:

1. Introduction to Windows programming.
2. The Visual C++ programming environment.
3. Object Oriented Programming and the MFC.
4. The document/view architecture and user interface elements.
5. Introduction to UNIX programming.
6. UNIX terminal I/O.
7. UNIX process management.
8. Interprocess Communication and Signals in UNIX.

Technical Systems Programming I | CSD305

COURSE NAME | COURSE CODE

III. LEARNING OUTCOMES AND ELEMENTS OF THE PERFORMANCE:

A. Learning Outcomes:

		Approx. % of Course Grade
1.	Discuss and apply programming concepts in the Windows environment with emphasis on event driven systems and the windows message loop.	10%
2.	Become proficient in the use of the Developer Studio to create and manage programming projects and to become familiar with the documentation system.	10%
3.	Review object oriented programming concepts and principles and apply those to Windows programming using the MFC.	
4.	Discuss and apply the document-view architecture and implement the user interface elements of the Windows environment within a program.	30%
5.	Discuss architectural issues and apply programming concepts in the UNIX environment.	5%
6.	Discuss and apply the general concepts of terminal I/O in the UNIX environment.	10%
7.	Discuss and apply the techniques for creating and managing processes in the UNIX environment.	15%
8.	Discuss and apply the various techniques for interprocess communications in UNIX..	10%
		100%

B. Learning Outcomes and Elements of the Performance:

Upon successful completion of this course the student will demonstrate the ability to:

1. Discuss and apply programming concepts in the Windows environment with emphasis on event driven systems and the windows message loop. This section will emphasize the fundamentals of windows programming by studying and writing a windows program using straight ANSI C and the API (i.e.; without using the MFC).

   Elements of the performance:

   - discuss the following and then apply that knowledge to writing a simple windows application:
     - the differences between DOS and Windows programming,
     - implications of event driven systems,
     - program development and, in general the Windows environment from the programmers perspective.

   - discuss the general capabilities of the Windows 3.1 API, and the techniques for calling API functions and then apply that knowledge in the writing of a simple Windows application.
     - Understanding and applying the understanding of message loops, window procedures and message processing will constitute the bulk of this section.

2. Become proficient in the use of the Developer Studio to create and manage programming projects and to become familiar with the documentation system.

   Elements of the performance:

   - create and execute non-MFC based projects.
   - create and execute MFC based projects.
   - add, delete and modify projects.
   - edit, compile, link, debug and in general build projects in the Developer Studio environment.
   - utilise the on-line documentation to answer questions directly related to Windows programming problems.

3. Object Oriented Programming and the MFC.

Elements of the performance:

- review and apply the following OOP techniques and terms to MFC programming:
  - the ***class*** and instantiating ***objects*** of type ***class***.
  - object classes, attributes, behaviours and instances of classes
  - methods and messages
  - encapsulation
  - inheritance and polymorphism
- MFC
  - define classes and variables
  - add, modify, work with private, protected, public data and function members.
  - inherit from the MFC classes.
  - extend and modify MFC class behaviour via virtual functions.

4. **Discuss and apply the document-view architecture and implement the user interface elements of the Windows environment within a program.**

Elements of the performance: write programs that demonstrate knowledge of-->

- document/view architecture (the CDocument and CView classes, serialization, collection classes, GDI, mouse and keyboard message handling)
- resource editing tools
- connect resources to code using classwizard
- menus and bitmaps
- dialog boxes
- controls
- DDX and DDV

5. Discuss architectural issues and apply programming concepts in the UNIX environment.

Elements of the performance:

- discuss and apply knowledge of the following to the writing of UNIX programs in C:
  - files
  - programs and processes
  - signals
  - process id's
  - groups
  - permissions
  - process attributes
  - interprocess communications

6. Discuss and apply the general concepts of terminal I/O in the UNIX environment.

Elements of the performance:

- discuss the general concepts of terminal I/O in UNIX and apply that knowledge in the writing of UNIX C programs.
- use ioctl in the writing of programs
- discuss and apply knowledge of the Curses screen management library to the writing of C programs.

7. Discuss and apply the various techniques for creating and managing processes in UNIX.

Elements of the performance:

- discuss and apply knowledge of the UNIX environment to write programs that manage processes with the fork, exit and wait system calls
- discuss and apply the various system calls for managing process ID's to writing programs with getgid, getegid, getpid, getpgrp, getppid, setuid, setgid, setpgrp.

8. Discuss and apply the various techniques for interprocess communications in UNIX.

Elements of the performance:

- write programs using pipes and FIFO's (named pipes)
- discuss and use semaphores for interprocess communication.
- discuss and apply shared memory and locks
- discuss and write programs using signals.
- discuss and use the kill, pause and alarm system calls in a C program.

IV. EVALUATION METHODS:

The mark for this course will be arrived at as follows:

Tests:

outcomes #1- #4	40%
outcomes #5 - #8	30%

Assignments:

outcomes #1 - #4	15%
outcomes #5 - #8	15%
Total	100%

The following letter grades will be assigned in accordance with the School of Engineering Technology and the School of Business and Hospitality policies:

Course Grading Scheme

A+	90% - 100%	consistently outstanding achievement
A	80% - 89%	outstanding achievement
B	70% - 79%	consistently above average achievement
C	55% - 69%	satisfactory or acceptable achievement in all areas subject to assessment
R	less than 55%	repeat - the student has not achieved the objectives of the course and the course must be repeated
CR		Credit Exemption
S		satisfactory given at midterm only
U		unsatisfactory given at midterm only
X		a temporary grade

An 'X' grade is limited to instances where exceptional circumstances have prevented the student from completing objectives by the end of the semester. An "X" grade must be arranged before the deadline for grade submission and is granted at the discretion of the Professor. The 'X' grade must also have the Dean's approval and has a maximum time limit of 120 days.

V. SPECIAL NOTES

1. In order to pass this course the student must obtain an overall **test** average of 55% or better, as well as, an overall **assignment** average of 55%.

2. Assignments must be submitted by the due date according to the specifications of the instructor. Late assignments will normally be given a mark of zero. Late assignments will only be marked at the discretion of the instructor in cases where there were extenuating circumstances.

3. The instructor reserves the right to modify the assessment process to meet any changing needs of the class. Consultation with the class will be done prior to any changes.

4. The method of upgrading an incomplete grade is at the discretion of the instructor, and may consist of such things as make-up work, rewriting tests, and comprehensive examinations.

5. Students with special needs (e.g. physical limitations, visual impairments, hearing impairments, learning disabilities) are encouraged to discuss required accommodations confidentially with the instructor.

6. Your instructor reserves the right to modify the course as he/she deems necessary to meet the needs of students.

VI. PRIOR LEARNING ASSESSMENT:

Students who wish to apply for advanced credit in the course should consult the instructor.

VII. REQUIRED STUDENT RESOURCES

Text:
- Using Visual C++, Gregory, Que books
- Visual C++ on-line documentation
- instructor supplied notes
- various magazine articles to be supplied