Subject Description Form

Subject Code	COMP3438
Subject Title	System Programming
Credit Value	3
Level	3
Pre-requisite / Co-requisite / Exclusion	**Pre-requisite**: COMP2432
Objectives	The objectives of this subject are to: 1. introduce students the concepts and principles of system programming and to enable them to understand the duties and scope of a system programmer; 2. provide students the knowledge about both theoretical and practical aspects of system programming, teaching them the methods and techniques for designing and implementing system-level programs; and 3. train students in developing skills for writing system software with the aid of sophisticated OS services, programming languages and utility tools.
Intended Learning Outcomes	Upon completion of the subject, students will be able to: *Professional/academic knowledge and skills* (a) organise the functionalities and components of a computer system into different layers, and have a good understanding of the role of system programming and the scope of duties and tasks of a system programmer; (b) grasp the concepts and principles, and be familiar with the approaches and methods of developing system-level software (e.g., compiler, and networking software); (c) apply the knowledge and techniques learnt to develop solutions to real-world problems; (d) select and make use of the OS kernel functions and their APIs, standard programming languages, and utility tools; (e) organise and manage software built for deployment and demonstration; and *Attributes for all-roundedness* (f) analyse requirements and solve problems using systematic planning and development approaches.

Subject Synopsis/ Indicative Syllabus

Topic
1. Introduction to System Programming and Unix Layered structure of a computer system; system software and application software; scope and tasks of system programming. Evolution of UNIX; features of UNIX; UNIX standards; good style of UNIX programming.
2. Introduction to UNIX Systems Files; types of UNIX files; UNIX file system; structure and representation of files in UNIX file system; directories; accessing files in UNIX; I/O redirection; devices and device drivers; UNIX file interface (APIs). UNIX shell; UNIX process creations and execution; process management; parent and child processes; UNIX process interfaces (APIs).
3. Introduction to Unix Device Driver Device Drivers; design issues; types of device drivers; major components of a device driver.
4. Device Driver Development OS/Driver interface; internal operations of a device driver; structure and major components; address spaces and data transfer; typical character/block driver design and implementation.
5. Overview of Compiler Construction Syntax and semantics of programming languages; language translation approaches; tasks of a compiler; the compiler process.
6. Lexical Analysis Tasks of lexical analysis; specifying tokens by regular grammars and regular expressions; recognizing tokens by Finite Automata (FA); construction of FA from regular expressions; converting NFA to DFA; simulating DFA.
7. Syntax Analysis Tasks of syntax analysis; specifying language constructs by context-free grammars; BNF; derivation; parse and syntax trees; recognizing language constructs by Pushdown Automata; top-down and bottom-up parsing methods.
8. Code Generation Intermediate compilation phases; symbol table; intermediate code generation; code optimisation; code generation.

Tutorials: 3 hours

Laboratory Experiment:

Topic
1. UNIX System and C Programming 2. UNIX Programming (processes, files, device drivers)

<table>
<tr><td><b>Teaching/ Learning Methodology</b></td><td>In lectures, concepts, models and algorithms will be explained with illustrative examples.

Tutorials and lab sessions help students understand concepts and improve their skills on solving problems.

Assignments help develop students' programming skills and critical thinking.</td></tr>
<tr><td><b>Assessment Methods in Alignment with Intended Learning Outcomes</b></td><td>
<table>
<tr><td rowspan="2">Specific assessment methods/tasks</td><td rowspan="2">% weighting</td><td colspan="6">Intended subject learning outcomes to be assessed</td></tr>
<tr><td>a</td><td>b</td><td>c</td><td>d</td><td>e</td><td>f</td></tr>
<tr><td><b>Continuous Assessment</b></td><td><b>55%</b></td><td colspan="6"></td></tr>
<tr><td>1. Assignments</td><td>35%</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>2. Mid-Term</td><td>20%</td><td>✓</td><td>✓</td><td>✓</td><td></td><td></td><td>✓</td></tr>
<tr><td><b>Examination</b></td><td><b>45%</b></td><td>✓</td><td>✓</td><td>✓</td><td>✓</td><td></td><td>✓</td></tr>
<tr><td>Total</td><td>100%</td><td colspan="6"></td></tr>
</table>
All three items are appropriate to evaluate the intended learning outcomes. Assignments are used to evaluate writing skills, critical thinking, and problem solving. Mid-term test and final examination can further help evaluate the related outcomes.</td></tr>
<tr><td><b>Student Study Effort Expected</b></td><td>
<table>
<tr><td colspan="2">Class contact:</td></tr>
<tr><td>▪ Lecture and Tutorial</td><td>39 Hrs.</td></tr>
<tr><td>▪ Lab</td><td>13 Hrs.</td></tr>
<tr><td colspan="2">Other student study effort:</td></tr>
<tr><td>▪ Assignments and Self-study</td><td>60 Hrs.</td></tr>
<tr><td>Total student study effort</td><td>112 Hrs.</td></tr>
</table></td></tr>
<tr><td><b>Reading List and References</b></td><td><b>Textbook:</b>

1. Aho, A.V., Lam, Monica S., Sethi, R. and Ullman, J.D., C<i>ompilers: Principles, Techniques, and Tools</i>, 2nd Edition, Addison-Wesley, 2006.

2. Molay, B., <i>Understanding Unix/Linux Programming</i>, Pearson Education, 2003.

<b>Reference Books:</b>

1. Stevens, W. R. and Rago, S. A., <i>Advanced Programming in the UNIX Environment</i>, 2nd Edition, Addison-Wesley, 2005.

2. Appel, A.W., <i>Modern Compiler Implementation in Java</i>, Foundation Books, 2007.</td></tr>
</table>

	3. Beck, L.L., *System Software: an Introduction to System programming*, 3rd Edition, Addison Wesley, 1996. 4. Cooper, K. and Torczon, L., *Engineering a Compiler*, Morgan Kaufmann, 2003. 5. Cooperstein, J., *Writing Linux Device Drivers: a guide with exercises,* CreateSpace, 2009. 6. Corbet, J., Rubini, A., and Kroah-Hartman, G., *Linux Device Drivers*, 3rd Edition, O'Reilly, 2005.