Testi del Syllabus

Resp. Did.	**CASELLI Stefano**	Matricola:	**004348**

Anno offerta:	**2016/2017**
Insegnamento:	**1002540 - SISTEMI OPERATIVI E IN TEMPO REALE**
Corso di studio:	**5015 - INGEGNERIA INFORMATICA**
Anno regolamento:	**2016**
CFU:	**9**
Settore:	**ING-INF/05**
Tipo Attività:	**B - Caratterizzante**
Anno corso:	**1**
Periodo:	**II° semestre**
Sede:	**PARMA**

Testi in italiano

Lingua insegnamento	Italiano
Contenuti	Sistemi operativi e in tempo reale: Programma del corso Architetture dei sistemi operativi Componenti dei sistemi operativi Programmazione concorrente nel modello a memoria condivisa Programmazione concorrente nel modello a memoria locale Sistemi di distribuiti e cliente-servitore Sistemi in tempo reale Schedulazione in tempo reale Programmazione di sistemi multiprocesso in C/C++ nei sistemi operativi UNIX e Linux. Programmazione multithread e in tempo reale con l'API POSIX in ambiente Linux.
Testi di riferimento	Agli studenti frequentanti sono rese disponibili sul sito del corso, lezione per lezione, le diapositive utilizzate in aula. I diversi argomenti affrontati nel corso sono trattati organicamente nei testi indicati di seguito: A. Silberschatz, P.B. Galvin, G. Gagne, “Sistemi operativi”, settima edizione o successiva, Pearson Education Italia, 2006 (o successivo). P. Ancilotti, M. Boari, “Programmazione concorrente e distribuita”, McGraw-Hill, 2007 J.W.S. Liu, Real-Time Systems, Prentice-Hall, 2000. D. Butenhof, Programming with POSIX Threads, Addison-Wesley, 1997.
Obiettivi formativi	Il corso si propone di far conoscere allo studente le architetture e le funzionalità dei moderni sistemi operativi e i concetti principali dei sistemi concorrenti e in tempo reale, e in particolare: - componenti dei sistemi operativi, processi, thread, spazi di indirizzamento, - modalità di gestione di processi e thread e di assegnazione di risorse del sistema, - meccanismi e strumenti per la sincronizzazione e la programmazione concorrente, - caratteristiche dei sistemi in tempo reale e relativi algoritmi di scheduling.

	Al termine del corso lo studente sarà in grado di applicare direttamente le conoscenze acquisite nei seguenti ambiti: - programmazione di applicazioni di sistema, multiprocesso e cliente-servitore in linguaggio C/C++ nei sistemi operativi UNIX e Linux, - programmazione di applicazioni multithread mediante l'API POSIX in Linux, - progettazione e programmazione di applicazioni multithread in tempo reale mediante l'API POSIX in Linux.
Prerequisiti	Conoscenza di base degli elementi dei sistemi operativi. Conoscenza generale della programmazione e capacità pratica di programmazione in linguaggio C/C++.
Metodi didattici	Lezioni in aula con proiezione di diapositive, integrata da una significativa attività guidata di laboratorio (obbligatoria) su programmazione multiprocesso, concorrente, multithread e real-time. Il corso si articola in circa 58 ore di lezioni in aula e 22 ore di esercitazioni assistite in laboratorio. La frequenza delle esercitazioni di laboratorio è obbligatoria. Sono proposte prove intermedie ed assegnamenti pratici o teorici da svolgere a casa per mantenere gli studenti al passo ed esonerarli da parti dell'esame.
Altre informazioni	Portale per il sito del corso: http://elly.dii.unipr.it Il materiale didattico e di supporto è reso disponibile sul sito del corso lezione per lezione agli studenti frequentanti.
Modalità di verifica dell'apprendimento	L'apprendimento viene verificato mediante una serie di prove pratiche, scritte e orali, in parte proposte come prove intermedie durante l'erogazione del corso. L'esame è superato solo quando tutte le prove sono state svolte con esito positivo: 1) Prova pratica di programmazione multiprocesso (preferibilmente da assumere durante il corso come prima prova intermedia); 2) Assegnamento pratico di programmazione multithread e real-time mediante Pthreads (proposto durante le esercitazioni obbligatorie); 3) Prova scritta di teoria della schedulazione real-time (seconda prova intermedia); 4) Prova scritta o pratica di programmazione concorrente; 5) Prova orale finale. La prova orale deve essere la prova conclusiva dell'esame. Per chi non sostiene o non supera le prove in itinere, le prove scritte n. 3) e 4) possono sostenute anche assieme negli appelli ufficiali. Ulteriori prove, inoltre, consentono di sostenere le prove pratiche n. 1) e 2) per chi non le ha superate durante il corso. La prova pratica n. 1) consente di verificare l'apprendimento delle competenze applicative di capacità di programmazione di sistema e di progettazione di applicazioni multiprocesso e cliente-servitore. La prova pratica n. 2) consente di verificare l'apprendimento delle competenze applicative di progettazione di applicazioni multi-thread e in tempo reale. La prova scritta n. 3) consente di verificare le conoscenze acquisite sui sistemi in tempo reale e i relativi algoritmi di schedulazione. La prova scritta o pratica n. 4) consente di verificare le conoscenze acquisite sull'uso delle tecniche di programmazione concorrente. La prova orale n. 5) consente di verificare complessivamente le conoscenze acquisite sui componenti dei sistemi operativi e sui fondamenti della programmazione concorrente.
Programma esteso	__Sistemi operativi e in tempo reale: Programma dettagliato del corso__ Architetture dei sistemi operativi Evoluzione, ruolo e funzioni del sistema operativo. Multiprogrammazione. Processi, thread e spazi di indirizzamento. Protezione. Commutazione di thread e di processi. Sistemi a macchine virtuali. Componenti dei sistemi operativi Gestione dei processi e dei thread. Scheduling della CPU. Gestione della

memoria. Memoria Virtuale. Gestione dell'I/O. Gestione dei file. Sicurezza. Tecniche di prevenzione e gestione del blocco critico.

Programmazione concorrente nel modello a memoria condivisa
Mutua esclusione e primitive di sincronizzazione. Gestione di risorse mediante semafori. Regioni critiche condizionali. Monitor. Supporti per la programmazione concorrente in Java.

Programmazione concorrente nel modello a memoria locale
Primitive send e receive. Remote procedure call. Comandi con guardia.

Sistemi distribuiti
Modello cliente-servitore. RPC in ambiente distribuito. Sincronizzazione, mutua esclusione e coordinamento distribuiti.

Sistemi in tempo reale
Sistemi embedded. Sistemi di elaborazione operanti con vincoli temporali. Tipologie dei sistemi in tempo reale e parametri caratteristici. Modello di riferimento per i sistemi di elaborazione in tempo reale.

Schedulazione in tempo reale
Scheduling di task aperiodici. Scheduling di task periodici mediante executive ciclico. Scheduling di task periodici basato su priorità. Algoritmi di scheduling Rate Monotonic ed Earliest Deadline First. Scheduling congiunto di task periodici, aperiodici e sporadici. Protocolli di accesso a risorse condivise. Gestione della inversione di priorità.

Programmazione di sistemi multithread e in tempo reale
Funzionalità dei moderni sistemi operativi a supporto dell'elaborazione in tempo reale. Supporti per il multithreading in Linux. La API standard POSIX. Thread e processi. Sincronizzazione tra thread. Segnali. Meccanismi di IPC. Scheduling. Gestione del tempo. Gestione della memoria. Gestione dell'I/O. Pattern per la programmazione di sistemi in tempo reale. Sistemi operativi real-time dedicati. Middleware per sistemi in tempo reale distribuiti.

Attività d'esercitazione
Esercitazioni in laboratorio relative alla programmazione di sistema multiprocesso in C/C++ nei sistemi operativi UNIX e Linux.
Esercitazioni in laboratorio con uso dell'API POSIX per la programmazione multithread e real-time in ambiente Linux.

Testi in inglese

Lingua insegnamento	Italian

Contenuti	Real-Time and Operating Systems: Course Syllabus Operating systems architectures Operating system components Concurrent programming in the global memory model Concurrent programming in the message passing model Distributed and client-server systems Real-time systems Real-time scheduling System and multiprocess programming in C/C++ in UNIX and Linux Multithread and real-time programming with the POSIX API in Linux
Testi di riferimento	Lecture notes are made available on the course web site to registered students. The following textbooks cover the topics addressed in the course: A. Silberschatz, P.B. Galvin, G. Gagne, “Operating Systems Concepts”, 7th edition or later, Addison-Wesley, 2006 (or later). (Java version also ok). P. Ancilotti, M. Boari, “Programmazione concorrente e distribuita”, McGraw-

	Hill, 2007 (in Italian). Alternative books in English on concurrent programming: G.R. Andrews, “Foundations of Multithreaded, Parallel, and Distributed Programming,” Addison-Wesley, 2000. S.J. Hartley, "Concurrent Programming - The Java Programming Language", Oxford University Press, 1998. J.W.S. Liu, "Real-Time Systems", Prentice-Hall, 2000. D. Butenhof, "Programming with POSIX Threads", Addison-Wesley, 1997.
Obiettivi formativi	Course learning objectives are to provide students with knowledge and understanding of architectures and functions of modern operating systems and of concurrent and real-time systems, including the key concepts of: - operating system components, processes, threads, address spaces; - process, thread, and system resource management; - synchronization mechanisms and concurrent programming techniques; - real-time systems and related scheduling algorithms. At the end of the course, students will be able to directly apply the learned techniques in the following contexts: - programming of system, multiprocess, and client-server applications in C/C++ in UNIX and Linux operating systems; - programming of multithread applications with the POSIX API in Linux; - design and programming of multithread real-time applications with the POSIX API in Linux.
Prerequisiti	Basic knowledge of operating systems concepts and working knowledge of C/C++ programming language is assumed.
Metodi didattici	Lectures in classroom with slide support, plus significant and mandatory supervised laboratory activity covering multiprocess, concurrent, multithread, and real-time programming. The course includes about 58 hours of classroom lectures and 22 hours of supervised laboratory activity. Attendance of laboratory classes is mandatory. During the semester, midterms and practical or theory home assignments are administered to keep students on pace with the course and exempt them from parts of the final exam.
Altre informazioni	Course web site available at: http://elly.dii.unipr.it for registered students. Teaching material is published on the web site as course progresses.
Modalità di verifica dell'apprendimento	Exam consists of a series of practice, written, and oral tests, including tests administered as midterms during the semester. Exam is passed when all individual tests have been passed: 1) Practice test on multiprocess programming (preferably to be assumed as midterm test); 2) Practice assignment on multithread and real-time programming using Pthreads (assigned during mandatory supervised laboratory classes); 3) Written test on real-time scheduling theory (second midterm); 4) Written or practical test on concurrent programming; 5) Final oral exam. Oral exam must be the final test. Written tests 3) and 4) can be taken together in the official exam sessions for those student which have not passed them in the midterms. Additional test dates are proposed also for the practice tests 1) and 2). However, it is strongly recommended that students engage in classroom and lab activities and pass the midterm and practice tests during the lecturing period. Practice test n. 1) assesses the degree of learning of practical skills concerning system programming and design of multiprocess and client-server applications. Practice test n. 2) assesses the degree of learning of practical skills concerning the design of multithread and real-time applications. Written test n. 3) assesses the degree of learning of knowledge concerning real-time systems and their scheduling algorithms. Written or practice test n. 4) assesses the degree of learning of knowledge

concerning concurrent programming techniques and their application.
Oral test n. 5) assesses the degree of learning of knowledge concerning operating system components and fundamental concepts of concurrent programming.

Programma esteso

__Real-Time and Operating Systems: Detailed Course Syllabus__

Operating systems architectures
Evolution, role and functions of the operating system. Multiprogramming. Processes, threads, and address spaces. Protection. Thread and process switch. Virtual machine systems.

Operating system components
Process and thread management. CPU scheduling. Memory management. Virtual memory. I/O management. File system. Security. Deadlock prevention and management.

Concurrent programming in the global memory model
Mutual exclusion and synchronization primitives. Resource management with semaphores. Conditional critical regions. Monitors. Concurrent programming in Java.

Concurrent programming in the message passing model
Send and receive primitives. Remote procedure call. Guarded commands.

Distributed systems
Client-server model. RPC in distributed environments. Distributed mutual exclusion, synchronization, and coordination.

Real-time systems
Embedded systems. Computer systems operating under real-time constraints. Real-time systems classification and relevant parameters. Reference model for real-time computer systems.

Real-time scheduling
Aperiodic task scheduling. Cyclic executive scheduling of periodic tasks. Priority-driven scheduling of periodic tasks. Rate-Monotonic and Earliest Deadline First scheduling algorithms. Priority-driven scheduling of mixed aperiodic, sporadic, and periodic tasks. Shared resources access control policies. Priority inversion management.

Multithreaded and real-time programming
Real-time computing support in modern, general-purpose operating systems. Multithreading support in Linux. The POSIX API standard. Threads and processes. Thread synchronization. Signals. IPC mechanisms. Scheduling. Time manangement. Memory management. I/O management. Patterns for real-time systems programming. Dedicated real-time operating systems. Middleware for distributed real-time computing systems.

Laboratory activities
Fundamentals of system programming in C/C++ in UNIX and Linux. Client-server multiprocess applications using various IPC mechanisms. Multithread and real-time programming with the POSIX API in Linux.