Corso di Laurea in Ingegneria Informatica (L8)
Anno Accademico 2016/2017

SISTEMI OPERATIVI

Docenti titolari dell'insegnamento (in ordine alfabetico): Prof. Salvatore Cavalieri (Canale 2) e Prof. Lucia Lo Bello (Canale 1)

Edificio/Indirizzo: Dipartimento di Ingegneria Elettrica Elettronica e Informatica, Via Santa Sofia, 64 - I95100 Catania, Edifici 3, 13

Telefono e Email Prof. Salvatore Cavalieri: +39-095-7382362, salvatore.cavalieri@unict.it

Telefono e Email Prof.ssa Lucia Lo Bello: +39-095-7382386, lobello@unict.it

Orario ricevimento: **Giovedì 15-17**

OBIETTIVI FORMATIVI	Acquisire la conoscenza dei concetti di base inerenti il progetto di sistemi operativi e la stesura di programmi in ambiente Linux/UNIX. Alla fine del corso gli allievi: -Conosceranno la struttura dei Sistemi Operativi, le relative problematiche di progetto e le politiche utilizzate per la virtualizzazione e per la gestione delle risorse (CPU, memoria centrale, memoria di massa, periferiche). - Acquisiranno conoscenze sui concetti di processo e di thread e sulla loro gestione. - Acquisiranno conoscenze sulle tecniche di gestione della concorrenza su risorse mutuamente esclusive. -Acquisiranno familiarità nell'interazione con la shell di Linux. Gli allievi a fine corso saranno in grado di scrivere applicazioni contenenti system call per: -creazione e gestione di processi, invio/gestione segnali, interazione e comunicazione tra processi; - gestire la concorrenza sulle risorse condivise; -creare applicazioni multithread.
REQUISITI	Requisiti necessari ad affrontare con successo gli argomenti del corso: -Conoscenza generale dell'architettura del calcolatore, in particolare: CPU, interruzioni, registri, memorie, tipologie di architetture multiprocessore, dispositivi di I/O. -Sulla programmazione: "familiarità con l'uso dei parametri argc e argv nella funzione main()" "uso dei puntatori e dei vettori in ANSI C" "uso dei vettori di strutture in ANSI C" "implementazione di liste, pile e code in ANSI C" "programmazione con funzioni e passaggio di parametri per valore e per riferimento" "puntatori a funzione" "tabelle hash".
FREQUENZA	La frequenza è fortemente consigliata sia per le prove in itinere sia per

LEZIONI	sostenere la prova d'esame.	
TESTI DI RIFERIMENTO	Per la parte teorica: Abraham **Silberschatz**, Peter Baer Galvin, Greg Gagne, "Sistemi Operativi, Concetti e esempi", Nona Edizione, Pearson, ISBN 9788865183717. In alternativa: Andrew S. **Tanenbaum**, "I moderni sistemi operativi 4/Ed.", 2016, ISBN 9788891901019. Ad integrazione della parte teorica e per tutta la parte pratica: R. Stones, N. Matthew, "**Beginning Linux Programming**", 4th edition, Wrox Press, 2007. Disponibile in rete	
PROVA D'ESAME	**PROGETTI E/O ELABORATI**	Non previsti.
	PROVE IN ITINERE	Sono previste due prove in itinere. Il voto finale risulta dalla media della valutazione delle due prove. Prova orale facoltativa se il voto finale è superiore a 18/30. Prova orale obbligatoria nel caso in cui una delle due prove sia superata pienamente (ossia, con voto maggiore di, o uguale a, 18/30) e l'altra con riserva (ossia, con voto superiore a 15/30, ma inferiore a 18/30). Le prove in itinere sono facoltative e si svolgono una dopo circa i 2/3 del corso (UDE 1, 2, 3) e una a fine corso (UDE 3,4,5).
	APPELLI	Gli appelli sono fissati di concerto con il CdS in ottemperanza alle modalità stabilite dal calendario accademico. L'esame si compone di una prova scritta, seguita da una prova orale. La prova scritta è rappresentata da un elaborato al calcolatore e da una sezione teorica, entrambe obbligatorie. La prova scritta può essere: non superata, oppure superata con riserva (se il voto è superiore a 15/30, ma inferiore a 18/30), o pienamente superata se il voto è maggiore di, o uguale a 18/30. La prova orale è obbligatoria per chi supera con riserva, mentre è facoltativa nel caso in cui la prova scritta sia pienamente superata e può essere svolta in un qualunque appello dell'anno accademico in cui la prova scritta è stata superata (pienamente o con riserva), fissato o concordato dal/con i Docenti.
	DATE D'ESAME	Pubblicate su https://www.ing.unict.it/, www.dieei.unict.it e sul portale di ateneo www.unict.it

	MODALITÀ DI PRENOTAZIONE	Prenotazione obbligatoria tramite portale di Ateneo. La data di chiusura delle prenotazioni è fissata di norma uno-due giorni prima dell'esame per questioni logistiche (il numero di postazioni limitate per la prova scritta al calcolatore richiede la necessità di programmare più turni nello stesso appello).
MATERIALE DIDATTICO	I testi si trovano in libreria, e in rete in formato elettronico. Dispense Docente, scaricabili dal sito studium.unict.it o dai siti personali dei Docenti (http://utenti.dieei.unict.it/users/scava/so270.html, http://lobello.dieei.unict.it/so1617). Possono riguardare sintesi e approfondimenti di teoria non disponibili nei libri di testo o codice C utilizzato per le esercitazioni.	

PROGRAMMA DEL CORSO

Gli argomenti in grassetto sono necessari all'acquisizione delle competenze e delle capacità minime.

ARGOMENTO	ORE	RIFERIMENTI
UNITÀ DIDATTICA 1: EVOLUZIONE, STRUTTURA E CLASSIFICAZIONE DEI SISTEMI OPERATIVI. IL SISTEMA GNU/LINUX. VIRTUALIZZAZIONE. • **Generalità sui sistemi operativi. Gestione delle risorse. Interfaccia utente. Concetto di Kernel.** • **Il sistema GNU/Linux.** • *Software libero e relative licenze d'uso. GPL.* • **Struttura dei sistemi operativi: monolitici, microkernel, ibridi, client/server.** • *La storia di Unix. La genealogia dei sistemi operativi. Lo standard POSIX. Peculiarità di Unix e differenze tra Unix e Linux.* • **Concetto di System call e passi necessari a realizzarle. Panoramica delle system call di Unix/Linux, Windows.** • *Classificazione dei sistemi operativi. Scelte progettuali relative alle diverse tipologie di SO.* • *Sistemi operativi per architetture multiprocessore.* • *Virtualizzazione. Macchine virtuali. Hypervisor di livello 1 e 2. Esempi: struttura di WMware e Virtual Box.*	8	Silbershatz: Cap.1, 2, 16 Tanenbaum: Cap.1, Cap.8.1.2 (multiprocessore) Dispense Docente

UNITÀ DIDATTICA 2: PROCESSI E THREAD • **Concetto di Processo. Diagramma a stati di un processo.** • **Interruzioni hardware e software e loro gestione nei sistemi operativi.** • **System call per creazione e gestione di processi: fork(), wait(), waitpid(). Famiglia delle exec().** • *Cenni di system call per la gestione di processi in ambiente Windows.* • **Segnali e loro gestione. System call kill(), signal(), alarm(), sigaction().** • **Esempi di programmi che impiegano tutte le system call menzionate.** • **Thread. Generalità, caratteristiche.** • **Implementazione dei thread: user space, kernel space, ibride (con riferimento alle scelte progettuali dei principali SO).** • *Attivazione dello Scheduler, upcall. Dati specifici dei thread. Gruppi di thread.* • **La libreria Pthread. Funzioni pthread_create(), pthread_join(), pthread_exit(), pthread_detach, pthread_attr_init().** • *Cenni di thread in ambiente Windows.* • **Cancellazione di thread.** • **Esempi di programmi che usano la libreria Pthread.** • **Linux: Esercitazione sui comandi di shell.** *Programmazione della shell.* • **Esercitazioni sui contenuti della UDE.**	16	Silbershatz: Cap.3, Cap.4 Tanenbaum: Cap.2.1, Cap. 2.2 Beginning Linux Programming: Cap. 2 (shell), Cap.11, Cap.12 Dispense Docente
UNITÀ DIDATTICA 3: GESTIONE DELLA CONCORRENZA, INTER PROCESS COMMUNICATION, DEADLOCK. • **Sezione critica. Mutua esclusione con attesa attiva. Semafori.** • **Problema del produttore e del consumatore e sua soluzione tramite semafori.** • **Mutex. Implementazione con thread in user space. Funzioni pthread_mutex.** • **IPC in Linux. Semafori: semget(), semop(), semctl(). Code di messaggi: msgget(), msgsnd(), msgrcv(), msgctl(). Shared memory: shmget(), shmat(), shmdt(), shmctl(). Esempi di programmi.** • **Comunicazione nei sistemi client-server: socket, pipe, FIFO.** • **Deadlock. Definizione del problema e strategie di gestione.** *Algoritmo del banchiere con risorsa singola.*	16	Silbershatz: Cap.3 (IPC, Socket, Pipe), Cap.5, Cap.7 (Deadlock) Tanenbaum: Cap.2.3, Cap.2.5, Cap.6 (Deadlock). Beginning Linux Programming: Cap. 12 (mutex), Cap.13, Cap. 14 (semafori, shared memory, code di messaggi), Cap 15. Dispense Docente

UNITÀ DIDATTICA 4: SCHEDULAZIONE DELLA CPU • **Schedulazione della CPU. Obiettivi. Algoritmi classici: FIFO, Round-robin, Schedulazione basata su priorità. Il problema della Starvation. Code multiple.** • *Cenni sullo scheduling nei sistemi real-time.* • *Cenni sullo scheduling di Linux: Completely Fair Scheduler.*	4	Silbershatz: Cap.6, Cap.7 Tanenbaum: Cap.2.4 Dispense Docente
UNITÀ DIDATTICA 5: GESTIONE DELLA MEMORIA CENTRALE, MEMORIA DI MASSA E I/O • **Cenni di gestione della memoria centrale. Spazio degli indirizzi. Segmentazione, Paginazione e Swapping.** • **File System. Gestione dello spazio su disco, affidabilità e prestazioni. System call sui file.** • **I/O Software.**	6	Silbershatz, Cap.8, Cap 11, 13 Tanenbaum, Cap.3, Cap.4 (File system), Cap.5 (I/O) Dispense Docente

COMPETENZE MINIME NECESSARIE AL SUPERAMENTO DELL'ESAME
Parte teorica Comprensione del concetto di sistema operativo e del suo ruolo in un sistema di elaborazione. Comprensione del concetto di kernel. Comprensione del concetto di interfaccia. Conoscenza delle diverse strutture dei sistemi operativi. Comprensione dei concetti di processo e di thread, con le relative differenze. Comprensione degli interrupt hardware e software e della loro gestione. Comprensione dei diversi modelli di multithreading. Comprensione del concetto di sezione critica, mutua esclusione e della gestione della concorrenza. Conoscenza del concetto di semaforo e di mutex. Conoscenza dei meccanismi di interprocess communication. Comprensione del problema della schedulazione della CPU. Conoscenza dei principali algoritmi di scheduling. Comprensione del problema della gestione della memoria centrale. Comprensione della struttura di un file system. Conoscenza di I/O software. **Parte pratica:** Capacità di realizzare programmi in cui due o più processi:

- si sincronizzano nell’accesso a risorse condivise,

- comunicano tra loro, tramite meccanismi di interprocess communication,

- scambiano segnali,

- lavorano su file condivisi.

Capacità di realizzare programmi multithread utilizzando la libreria pthread.

Capacità di utilizzo dei comandi della shell di Linux.

ESEMPI E MODELLI DI DOMANDE E/O ESERCIZI
Esempi e modelli sono disponibili sul sito dell’insegnamento