Esame di Sistemi Operativi – A.A.2014-2015

Il sito web del corso è www.units.it/mumolo ed e' disponibile all'indirizzo

Didattica → Corso di Sistemi Operativi, laurea triennale in Ingegneria dell'Informazione → AA 2014/2015

Il materiale – sotto forma di slide pdf e dispense - è scaricabile dal sito all'indirizzo Presentazioni del corso .

Ogni studente deve iscriversi al corso dal sito moodle2.units.it

Se lo studente vuole consultare altro materiale, può leggere i classici testi (disponibili alla libreria tecnico-scientifica) tra cui si consiglia:
Modern operating systems / Andrew S. Tanenbaum.

Sistemi operativi / A. Silberschatz, P. Galvin.

L'esame di S.O. di questo AA si articola in tre provette svolte durante l'anno e che riguardano gli aspetti svolti a lezione. Lo studente deve poi presentare all'esame una serie di esercizi proposti durante l'anno e svolti a sua discrezione. I sorgenti degli esercizi verranno raggruppati in un file archivio e mandati a mumolo@units.it.

Il punteggio finale verr'à assegnato come media pesata delle provette e degli esercizi, come qui indicato:

voto_1a_provetta*0.3 + voto_2a_provetta*0.3 + voto_3a_provetta*0.3 + voto_esercizi * 0.1

Ogni provetta contribuisce quindi al voto finale per il 30% e gli esercizi per il 10%.

Ci sono due alternative alla modalità provette: scritto+orale oppure tesina+orale.

La tesina deve essere discussa con il docente

Nel caso scritto + orale la valutazione finale e' data dalla media dei due voti. Nel caso tesina+orale il voto finale è dato da:

$$\text{Voto\_Tesina} * \alpha + (1 - \alpha) \cdot \text{Voto\_Orale}$$

Infatti, ad ogni tesina verrà assegnato un peso (α) relativo alla difficoltà della tesina stessa.
Orale e tesina vengono valutate in trentesimi.

In ogni modo la realizzazione degli esercizi (che verranno indicati sul sito) è necessaria per la registrazione dell'esame.

Programma dettagliato

Introduzione al corso-sito web-materiale-esame-testi di consultazione

funzioni-utilità-uso
punti di vista di un SO: macchina astratta-gestore risorse-algoritmi di gestione interna
modalità sistema-utente
Concetti generali:risorse/programma/processi-programmi/thread/contesto dei processi/diagramma degli stati
Spooling
il kernel e la sua storia: organizzazione interna
PCB - code di processi-tabella processi-IPC
schedulazione – context switch – time sharing
cenno alla implementazione di uno schedulatore TS

Modelli della concorrenza mediante code attesa

Concorrenza
-generalità, interleaving
-grafi alle precedenze
-strumenti linguistici(coroutine,fork-join,cobegin-coend)
-sincronizzazione
-determinatezza

Java come ambiente di programmazione concorrente

Programmazione concorrente: implementazione-esempi

Programmazione concorrente: costrutti linguistici
-semafori:uso nella sincronizzazione e soluzione della mutua esclusione
-monitor: uso nella sincronizzazione e nella mutua esclusione

Programmazione concorrente: problematiche
-mutua escusione: esempi-soluzioni
-stallo: esempi-soluzioni

Programmazione concorrente: problemi classici

Linguaggio C

Programmazione concorrente in C

Il sistema operativo Linux:
-organizzazione del nucleo
-chiamate di sistema
-IPC
-identificatori

Gestione risorse:
-CPU
-memoria
-disco
-file system

-file system/protezione file
-bufferizzazione in memoria
-comandi utente
-programmazione di shell
-programmazione di sistema in C
-compilazione del kernel
-moduli del kernel
-device driver

Tesine proposte

Vengono ora proposte alcune tesine da portare all'esame nel caso lo studente scelga la modalità **tesina+orale**. Questo elenco non è vincolante, serve solo a dare dei suggerimenti.

1 Problemi di programmazione in Bash Shell

1. Scrivere uno script in Bash per la gestione testuale del comando mail: analizzando il file /var/mail/#utente# produrre i seguenti risultati: spedire un messaggio a una o pi´u persone, visualizzare i mail per oggetto o per mittente, lista dei messaggi dal pi´u nuovo al pi´u vecchio, reindirizzare alcuni messaggi ad un altro indirizzo, cancellare i messaggi selezionati, salvare alcuni messaggi selezionati in un file.
(peso= 0.3)

2. Scrivere uno script in Korn Shell per gestire una agenda appuntamenti con le seguenti funzioni:
aggiungi appuntamento, visualizza appuntamenti, modifica appuntamenti esistenti,
cancellazione appuntamenti. Utilizzare il comando crontab.
(peso= 0.3)

2 Problemi di programmazione in Perl

1. Usando le primitive standard di IPC tramite messaggi, costruire il meccanismo semaforico e con esso risolvere il problema del produttore/consumatore con due consumatori creando un log di funzionamento.
(peso= 0.45)

2. Utilizzando le primitive di IPC basate sulla memoria condivisa e sui semafori, realizzare un programma che scambia messaggi (subroutine send-receive). Risolvere il problema del produttore/consumatore generando un file di log per verificare il funzionamento.
(peso= 0.5)

3. Realizzare uno script in Perl per risolvere il problema del produttore/consumatore con l’algoritmo di Peterson. Le variabili vengono condivise con la memoria condivisa. Generare una traccia di funzionamento.
(peso= 0.45)

4. Risolvere il classico problema dei 5 filosofi a cena con processi concorrenti e sincronizzazione semaforica, gemerando una traccia di funzionamento.
(peso= 0.4)

5. Risolvere il problema del barbiere. In un negozio di barbiere ci sono N

sedie riservate ai clienti e una sedia riservata al cliente che viene servito. Il barbiere o serve un cliente o dorme sulla poltrona riservata al cliente. Il cliente che entra o vede che tutte le sedie sono occupate, ed allora se ne va, o se il barbiere ´e occupato ma ci sono sedie disponibili, si siede ed attende che il barbiere si liberi o, infine, se il barbiere sta dormendo, lo sveglia ed inizia ad essere servito.
Risolvere il problema in Perl programmando il barbiere come server e i clienti come client.
(peso= 0.4)

7. Problema dell'incrocio. Una strada di grande viabilit´a incrocia con una strada pedonale, ed i passaggi sono regolati da un semaforo che ´e normalmente verde per la strada di grande viabilit´a. Se un pedone vuole passare preme un apposito pulsante. Realizzare in Perl una applicazione nella quale il semaforo ´e un processo server e i pedoni sono processi client. Un client invia una richiesta di passaggio e riceve la risposta corrispondente mediante uno scambio di messaggi con il server. Il semaforo accoglie le richieste dei pedoni se non ci sono richieste pendenti dei veicoli.
(peso= 0.5)

3 altro (da discutere con il docente)