

Corso di Laurea in Ingegneria Informatica

Sistemi Operativi A

a.a. 2002/03

Francesco Zanichelli

Recapiti

tel: 0521 - 905710 (interno: 5710)

fax: 0521 - 905723 (interno: 5723)

email: francesco.zanichelli@unipr.it

Orario di ricevimento: **Giovedì' 11:00 - 13:00 (Pal. 1)**

Programma di massima del corso

Teoria

- compiti di un Sistema Operativo
- multiprogrammazione e processi concorrenti
- modello a scambio di messaggi
- modello ad ambiente globale
- scheduling della CPU

UNIX

- interazione con l'utente: file system, shell e comandi
- programmazione di sistema:
 - primitive per gestione I/O e processi
 - sincronizzazione comunicazione attraverso:
 - segnali
 - pipe e fifo
 - socket

Testi consigliati

Come testo generale sui sistemi operativi

W. Stallings, "Sistemi Operativi", Jackson Libri, 2000

limitatamente ai cap. 1 ,2 3, 4 (solo 4.1) , 5 (solo 5.1, 5.2, 5.3, 5.4) , 6 (solo 6.1 e 6.7), 9

Il testo non è considerato indispensabile ma può essere utile ad approfondire e a collegare i concetti introdotti nella parte di teoria.

Possibili alternative

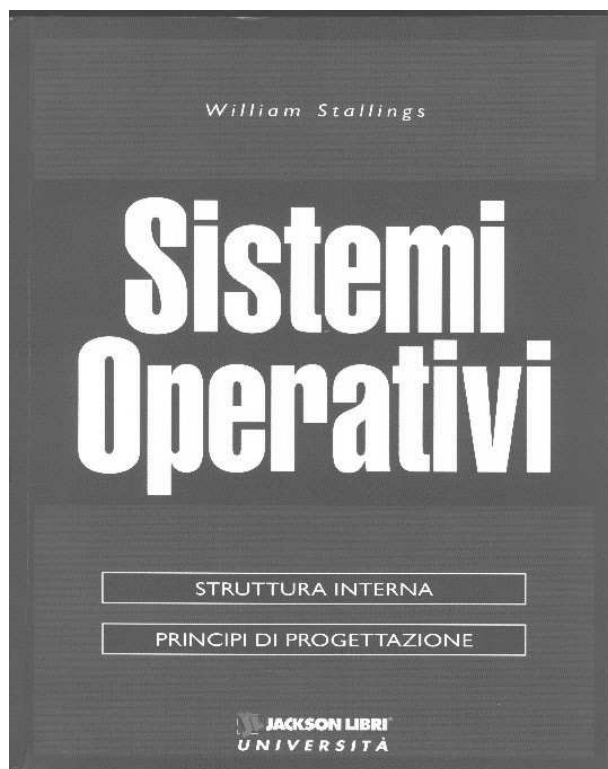
- A. Silbershatz, P. Galvin, G. Gagne, "Sistemi Operativi," Addison-Wesley, sesta ed. 2002.
- A.S. Tanenbaum, "I Moderni Sistemi Operativi," Jackson Libri, seconda ed., 2000.

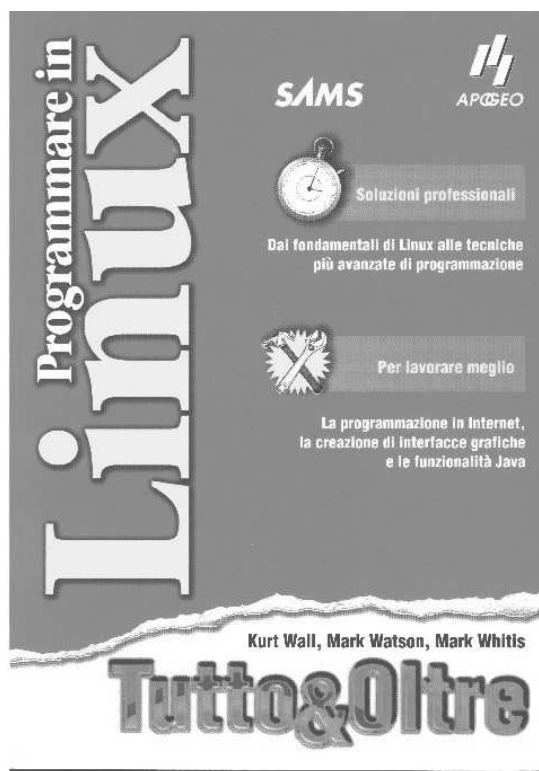
Per la programmazione di sistema in UNIX:

- K. Wall, M. Watson, M. Whitis, "Programmare in Linux – Tutto & Oltre", Apogeo, 2000

Possibili alternativa

- W.R. Stevens, "Advanced Programming in the UNIX Environment," Addison-Wesley, 1993.





Dispense

Sono disponibili copie elettroniche (in PDF dalla pagina del corso o direttamente <http://www.ce.unipr.it/~mczane/lucidi-sisopA.html>) e cartacee (presso il centro documentazione della sede scientifica) dei lucidi utilizzati a lezione

Iscrizione obbligatoria agli esami e ai compitini via Internet

<http://esami.ceda.unipr.it/esami/>

➡ obbligatorio per tutti iscriversi all'appello fittizio del 14/3/03 (per raccogliere l'elenco degli studenti del corso)

Modalità esame

Prova scritta in due parti:

- Esercizio in laboratorio su interazione tra processi UNIX
- 2 o 3 domande di teoria

E' possibile sostenere l'esame per parti, superando le due prove in appelli diversi.

Risorse utili su Internet

➡ www.ce.unipr.it/didattica/sisopA.html Pagina Web del corso (in perenne costruzione)

<http://gatil.firenze.linux.it/index.html> Guida alla Programmazione in Linux (© **Simone Piccardi**)

Unix Programming FAQ (le domande più frequenti riguardo alla programmazione UNIX)

Unix SOCKET FAQ (le domande più frequenti riguardo alla programmazione delle socket UNIX)

LINUX

- una versione completa e affidabile di UNIX
- disponibile per PC x86 Intel/AMD e numerose altre piattaforme
- strumento (quasi) indispensabile per le esercitazioni
- include gli strumenti di sviluppo necessari
 - compilatore C (gcc)
 - editor (vi , emacs, xemacs)
 - debugger (gdb, ddd)
 - manuali on-line (comandi e primitive)

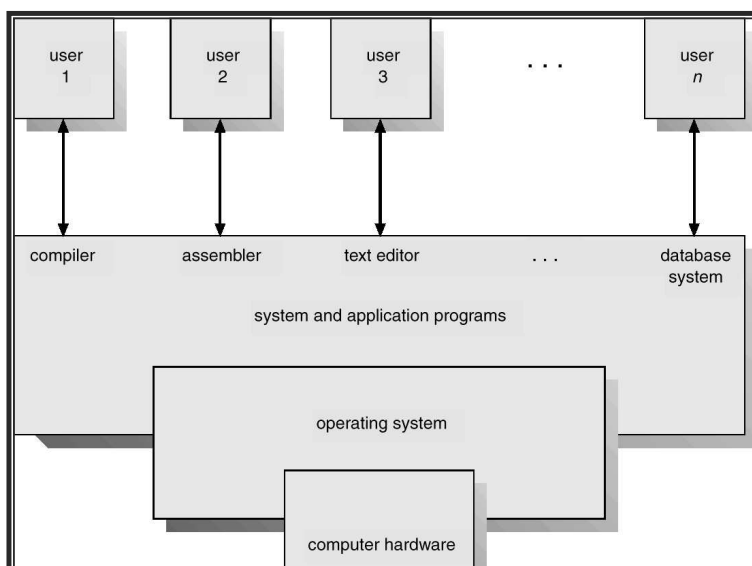
LINUX

- numerose distribuzioni (RedHat, Mandrake, Suse, Debian, ...)
 - può coesistere con Windows
 - dual-boot
 - partizioni separate o comuni
 - possibile ridimensionamento di una partizione Windows esistente (fips)
- numerose guide alla installazione
- ricercate con google “guida installazione Linux” – ad es.
<http://linux.html.it/guida>

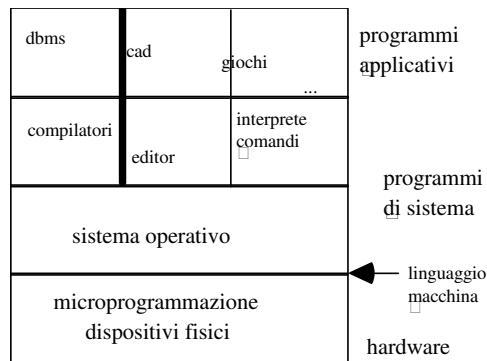
Introduzione

Un *sistema di elaborazione* puo' essere visto come l'insieme di:

- hardware
- sistema operativo
- programmi applicativi
- utenti



Sistema di elaborazione



Che cos'è un sistema operativo?

- Un sistema di calcolo può essere visto come un insieme di risorse Hw e Sw utilizzate per lo sviluppo e la esecuzione dei programmi di utente.
- Tali risorse devono essere:
 - utilizzate secondo *un determinato ordine*;
 - rese disponibili *a più utenti*;
 - *protette* contro accessi non autorizzati;
 - organizzate in modo da garantire la *sopravvivenza del sistema* in caso di guasti;
 - gestite in modo che risulti *semplificato ed efficiente il loro uso*, etc.
- Con il termine **sistema operativo** si intende quell'insieme di programmi che provvedono alla gestione delle risorse Hw e Sw di un sistema di calcolo.
- Una definizione alternativa (Tanenbaum):
un sistema operativo è un programma che *controlla le risorse* di un calcolatore e fornisce ai suoi utenti un'interfaccia o *macchina virtuale* più agevole da utilizzare della macchina "nuda".

Sistema Operativo

- puo` essere visto come:

- allocatore di risorse Hw e Sw:

tempo di CPU, spazio di memoria, dispositivi di I/O, compilatori, etc.

- le risorse devono essere assegnate a programmi specifici secondo determinate politiche.

- programma di controllo:

controlla l'esecuzione dei programmi per prevenire errori ed usi impropri del calcolatore (in particolare per il controllo dei dispositivi di I/O).

- obiettivi del S.O.:

- rendere *piu` semplice* l'uso di un sistema di calcolo

- rendere *piu` efficiente* l'uso delle risorse del sistema di calcolo.

Sistema Operativo

Il **sistema operativo** e` costituito dall'insieme dei programmi (software o firmware) che *rendono praticamente utilizzabile* l'elaboratore agli utenti cercando contemporaneamente di *ottimizzarne le prestazioni*.

- Visione top-down: il sistema operativo come una *macchina estesa* (fornisce astrazione, hiding di dettagli)

- Visione bottom-up: il sistema operativo come un *gestore di risorse* (fornisce protezione, risoluzione di conflitti o interferenze)

Risorse hardware

- processori (registri, unità aritmetiche, parallelismo interno)
 - memorie
 - canali di comunicazione
 - dispositivi di I/O
-
- Grande evoluzione sia sui singoli componenti che sulle tecniche di collegamento
-
- Spostamento di intelligenza verso i dispositivi
-
- Gerarchia di memoria
 - memoria centrale (principale, core)
 - memoria cache
 - memoria secondaria

Gestione delle risorse

Significa:

- tenere traccia delle risorse
- adottare strategie di assegnazione
- allocare le risorse
- recuperare le risorse inutilizzate
- rilevare eventuali usi impropri

Funzioni specifiche:

- gestione della memoria principale
- gestione dei processori
- gestione dei dispositivi periferici
- gestione della memoria secondaria

Funzioni specifiche di gestione

- gestione dei dispositivi periferici

- mascherare al programmatore la complessità delle operazioni di I/O
- effettuare controlli sul corretto funzionamento delle operazioni
- risolvere conflitti nell'utilizzo di una stessa periferica da parte di più programmi
- consentire il massimo sfruttamento delle periferiche.

- gestione dei processori

- decidere quale programma userà il processore (scheduling) in base a criteri di corretto funzionamento e di efficienza
- verificare che i programmi rilascino il processore entro il tempo stabilito.

Funzioni specifiche di gestione

- gestione della memoria centrale

- caricare in memoria programmi e dati
- evitare interferenze fra programmi diversi
- assegnare la memoria in base a criteri di efficienza
- minimizzare i trasferimenti tra memoria centrale e memoria di massa.

- gestione della memoria secondaria

- consentire l'accesso all'informazione in base alla sua organizzazione logica (**File System**) anzichè fisica (ad es. piatti, tracce, settori)
- controllare i diritti di accesso ai file da parte degli utenti
- consentire creazione, modifica e cancellazione dei file, ...

Proprieta` fondamentali di un S.O.

- affidabilita`
- efficienza

Aree di applicazione di un S.O.

- sistemi di tipo generale
- sistemi in tempo reale
 - applicazioni per il controllo di processo
 - applicazioni di tipo gestionale (interrogazione di basi di dati)

Funzioni di un S.O.

- definizione e gestione dell'interfaccia utente
- gestione dei lavori degli utenti
- gestione delle risorse del sistema
- ausilii per la messa a punto dei programmi
- ausilii per la gestione dei dati -- file system
- funzioni ausiliarie di sistema per
 - affidabilità
 - sicurezza
 - contabilità

Utenti del S.O.

- *utenti finali* del sistema
per essi il sistema operativo è *trasparente*
- *programmatori applicativi*
utilizzano i servizi del S.O. per la realizzazione e l'esecuzione dei loro programmi
- *programmatori di sistema*
aggiornano e modificano i programmi del S.O. per adeguarli a nuove necessità del sistema o degli utenti applicativi
- *operatori*
controllano il funzionamento e rispondono alle richieste di intervento da parte del sistema
- *amministratore del sistema*
stabilisce le politiche di gestione del sistema e ne cura l'osservanza

Tipi di S.O.

sistemi proprietari

- progettati dai costruttori al fine di *sfruttare in modo ottimale* le risorse di ciascun tipo di macchina
- l'interfaccia con l'utente varia tra le diverse famiglie di sistemi
- esempi:
 - IBM: OS/360 - 370, VM, MVS
 - DEC: RT-11, VMS
 - ...

sistemi standard

- progettati da case di software o da grandi utenti per *creare applicazioni portabili* su sistemi diversi
- l'interfaccia con l'utente rimane costante nelle diverse versioni
- esempi:
 - UNIX, MS-DOS, Windows

Evoluzione nell'uso dei calcolatori

- Scrivere programmi che realizzano algoritmi:
 - strutture dati => *transienti*
 - libreria di sottoprogrammi => *capitale*
- Evoluzione verso applicazioni in cui i dati rappresentano lo stato del sistema che evolve
 - strutture dati => *capitale*
 - le strutture dati sopravvivono al programma
- Embedded applications (sistemi bancari, banche dati, controllo di processo)
- Evoluzione verso applicazioni di A.I.:
 - riconoscimento del linguaggio naturale
 - basi di conoscenza, sistemi esperti
 - robotica
 - visione