Corso di Laurea in Ingegneria Informatica

Sistemi Operativi A

a.a. 2008/09

Francesco Zanichelli

Recapiti

tel: 0521 - 905710 (interno: 5710)

fax: 0521 - 905723 (interno: 5723)

email: francesco.zanichelli@unipr.it

Orario di ricevimento: **Giovedì 10:30 - 12:30 (Pal. 1 – Sede scientifica di Ingegneria)**

Programma di massima del corso

Teoria (prima parte del corso completata entro 16/04/09 - compitino il 28/04/09)

- compiti, funzioni e organizzazione di un Sistema Operativo
- multiprogrammazione e processi concorrenti
- modelli di interazione tra processi :
  - ambiente globale (sincronizzazione con semafori)
  - scambio di messaggi
- scheduling della CPU

UNIX

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

Testi consigliati

Come testo generale sui sistemi operativi:

- Silbershatz, Galvin, Gagne, "Sistemi Operativi - Concetti ed esempi" Pearson-Addison Wesley, 2006. (Settima Edizione), limitatamente a :
  - cap. 1,2,3 – tutto / cap. 4 - solo 4.1 / cap. 5 - tutto tranne 5.4 e 5.5
  - cap. 6 - 6.1, 6.2, 6.4. 6.5, 6.6.1 / cap 8 - 8.1 e 8.2 / cap 14 - leggere tutto

- P. Ancillotti, M. Boari, A. Ciampolini, G. Lipari, Sistemi Operativi, McGraw-Hill,Seconda edizione, 2007

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

Per la programmazione di sistema in UNIX:

GAPIL (Guida alla Programmazione in Linux) http://gapil.truelite.it/index.html di © Simone Piccardi (N.B. utilizzabile anche durante la prova di esame)

Testi di difficile reperibilità

- K. Wall, M. Watson, M. Whitis, “Programmare in Linux – Tutto & Oltre”, Apogeo, 2000
- W.R. Stevens, "Advanced Programming in the UNIX Environment," Addison-Wesley, 1993.

Dispense

Sono disponibili copie elettroniche in PDF dei lucidi utilizzati a lezione dal sito del corso.

Iscrizione obbligatoria agli esami e ai compitini via Internet

MODALITA' DI ESAME

Prova in itinere sulla parte di teoria il 28 Aprile (dalle 12.30 alle 14.30)

Prova scritta in due parti:

- esercizio in laboratorio su una interazione tra processi UNIX (in C/C++)
- 3 domande di teoria (per chi non ha sostenuto la prova in itinere)

Gli studenti possono sostenere il giorno dell'esame una sola o entrambe le prove.

I risultati parziali conseguiti rimangono validi per tutto l'anno accademico (sessione di gennaio/febbraio quindi compresa), ed e' possibile ripetere, al più una volta, una prova per migliorare il voto conseguito. In ogni caso si mantiene il voto migliore tra quelli conseguiti per ciascun tipo di prova (teoria o UNIX).

Per calcolare il voto finale, sarà eseguita una media pesata dei risultati ottenuti nei due tipi di prova (con pesi 0.6 per la prova UNIX e 0.4 per la prova di teoria).

Risorse utili su Internet

Sito del corso https://corsi.unipr.it/SOA0809

http://gapil.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 Fedora, Mandriva, Suse (in laboratorio), Debian, Ubuntu, LiveCD Knoppix,…)

Può coesistere facilmente con Windows in diverse modalità:

1. modalità **dual-boot** (scelta del S.O. all'avvio del sistema) con
   A. installazione di Linux in partizione separata (possibilità di ridimensionare una partizione Windows esistente per ottenere lo spazio per una nuova partizione Linux)
   B. installazione di Linux nella partizione di Windows ⇒ http://wubi-installer.org
2. modalità **macchina virtuale**

   installazione di un applicazione di virtualizzazione (Vmware Player, Virtualbox) e installazione/download di un macchina virtuale Linux
3. modalità **ambiente operativo POSIX in Windows**

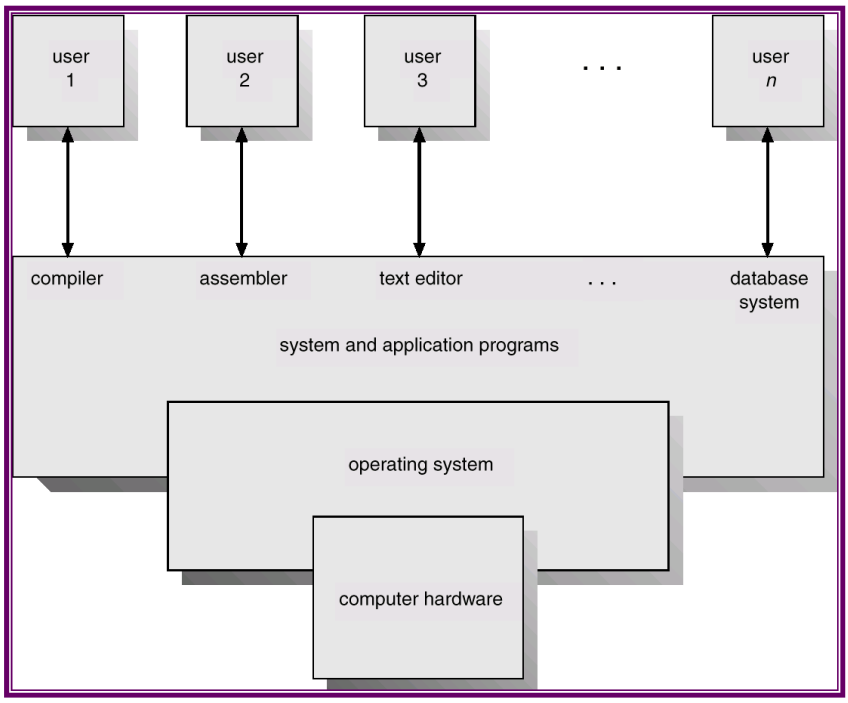
   ⇒ installazione dell'applicazione **CYGWIN** (http://www.cygwin.com)

Numerose guide all'installazione (HOWTO) ⇒ cercare con google "guida installazione Linux" – ad es. http://linux.html.it

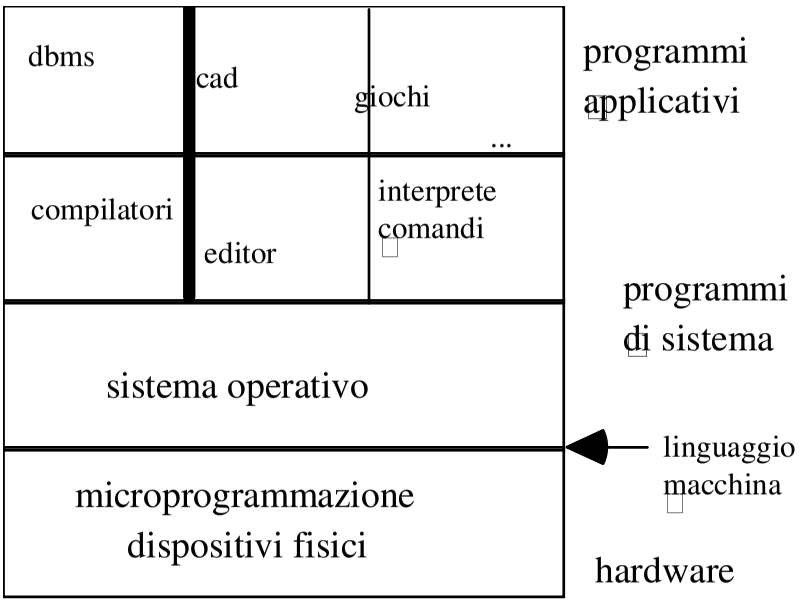
Introduzione

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

- hardware
- sistema operativo
- programmi applicativi
- utenti



Sistema di elaborazione



Che cos'e` un sistema operativo?

- Un sistema di calcolo puo` 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 piu` 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 e' un programma che *controlla le risorse* di un calcolatore e fornisce ai suoi utenti un'interfaccia o *macchina virtuale* piu' *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 (job, programmi) 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.

<u>sistemi proprietari</u>

- 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
  - ...

<u>sistemi standard</u>

- 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