

Corso di Laurea in Informatica

Sistemi Operativi (codice xxxxx - 6 CFU)

Anno Accademico 2011/12

Docente: Domenico Tegolo

Calendario delle lezioni

01) 10 - Ottobre	11:30/13:30	2	13) 22 - Novembre	09:30/11:30	2
02) 11 - Ottobre	09:30/11:30	2	14) 28 - Novembre	11:30/13:30	2
03) 17 - Ottobre	11:30/13:30	2	15) 29 - Novembre	09:30/11:30	2
04) 18 - Ottobre	09:30/11:30	2	16) 05 - Dicembre	11:30/13:30	2
05) 24 - Ottobre	11:30/13:30	2	17) 06 - Dicembre	09:30/11:30	2
06) 25 - Ottobre	09:30/11:30	2	18) 12 - Dicembre	11:30/13:30	2
07) 31 - Ottobre	11:30/13:30	2	19) 13 - Dicembre	09:30/11:30	2
08) 07 - Novembre	11:30/13:30	2	20) 19 - Dicembre	11:30/13:30	2
09) 08 - Novembre	09:30/11:30	2	21) 20 - Dicembre	09:30/11:30	2
10) 14 - Novembre	11:30/13:30	2	22) 09 - Gennaio	11:30/13:30	2
11) 15 - Novembre	09:30/11:30	2	23) 10 - Gennaio	09:30/11:30	2
12) 21 - Novembre	11:30/13:30	2	24) 16 - Gennaio	11:30/13:30	2

Monte Ore: 48

Crediti: 6

Appelli di Esami: 2-Estate
2-Autunno
2-Inverno

Come raggiungermi: tel. 09123891119

fax. 09123891024

domenico.tegolo@unipa.it

Date d'esame: Dopo il 6 Giugno
da concordare con gli studenti e
il docente di Laboratorio S.O.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

1

Sistemi Operativi Calendario delle Lezioni

Time	Lunedì	Martedì	Mercoledì	Giovedì	Venerdì
08:30-09:30					
09:30-10:30					
10:30-11:30		Sist. Oper. (II/III - L.T.)			
11:30-12:30					
12:30-13:30	Sist. Oper. (II/III - L.T.)				

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Chiarimenti sulle modalità di svolgimento del Corso

- **Il ricevimento studenti** deve essere utilizzato per problemi dei singoli studenti ed è inerente al programma di S.O.
- Se ci sono punti non chiari nella lezione, **non esitate a chiedere spiegazioni**.
- Poiché **non sono depositario della conoscenza assoluta**, la risposta POTREBBE essere rimandata ad una lezione successiva...
- Durante il corso si svolgerà **una prova scritta**, o mid-term relativa agli argomenti sino ad allora presentati, verosimilmente le memorie.
- Questa è una semplificazione per **incentivare lo studio durante l'anno** e identificare le proprie conoscenze in S.O. quindi non pregiudica l'esame finale.
- La prova mid-term è **riservata agli studenti dell'anno** accademico corrente.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Chiarimenti sulle modalità di svolgimento del Corso

Durante gli scritti

- **E' vietato comunicare** in qualunque modo (oralmente, in forma scritta o elettronicamente) e per qualsivoglia motivo.
- Chi viene sorpreso a parlare, viene **invitato a lasciare l'aula** e a ripresentarsi al prossimo appello.
- **Questo vale** sia per chi parla che per chi ascolta.
- **Il compito potrà essere annullato** anche in caso di manifesta copiatura scoperta nel corso della correzione dello scritto.
- Anche in questo caso, l'annullamento riguarda sia il "**copiatore**" che il "**copiato**".

Nota: Se avete bisogno di qualcosa, come una penna o un foglio, chiedete al docente

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Argomenti del Corso

Introduzione ai sistemi di elaborazione

Introduzione ai sistemi operativi

- ✓ Che cosa è un S.O.
- ✓ Cenni storici sui S.O.
- ✓ Classificazione dei S.O.
- ✓ Concetti base sui S.O.
- ✓ Chiamate di sistema

Processi e thread

- ✓ Introduzione ai processi
- ✓ Thread
- ✓ Comunicazione tra processi
- ✓ Problemi di comunicazione
- ✓ Schedulazione tra processi

Deadlock

- ✓ Introduzione ai Deadlock
- ✓ Identificare e risolvere dei deadlock
- ✓ Evitare i deadlock
- ✓ Prevenzione da Deadlock

Gestione della Memoria

- ✓ Sistemi per la gestione della memoria
- ✓ Swapping
- ✓ Memoria Virtuale
- ✓ Algoritmi di rilocazione delle pagine
- ✓ Segmentazione

Input/output

- ✓ Principi dell'hardware e del software
- ✓ I livelli software dell'I/O
- ✓ I Dischi
- ✓ I Clock

I File System

- ✓ I file
- ✓ Le directory
- ✓ Implementazione del File System

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

5

Sistemi Operativi

Testi Consigliati

Andrew S. Tanenbaum
I moderni Sistemi Operativi 3th Ed.
Pearson-Prentice Hall

Harvey M. Deitel, Paul J. Deitel, David R. Choffnes
Sistemi Operativi 3^a Ed.
A cura di William Fornaciari
Person-PrenticeHall

Abraham Silberschatz, Peter Baer Galvin
Sistemi Operativi 6nd Ed.
Addison - Wesley

William Stallings
Sistemi Operativi
Jacksonlibri

Introduzione ai Sistemi di Elaborazione

La **capacità di comprendere** le funzionalità di un Sistema Operativo sono subordinate alla **conoscenza** dell'organizzazione di un Sistema di Elaborazione.

Prima di esaminare i sistemi operativi è fondamentale disporre di alcune nozioni di base sul funzionamento dell'hardware del sistema.

A livello più astratto un computer o sistema di elaborazione è costituito dai seguenti moduli funzionali:

Processore: Controlla le operazioni del computer, quando questo è unico, spesso viene chiamato CPU.

Memoria Centrale: Memorizza dati e programmi, è tipicamente volatile.

Moduli di I/O: trasferiscono i dati fra il computer e il suo ambiente esterno.

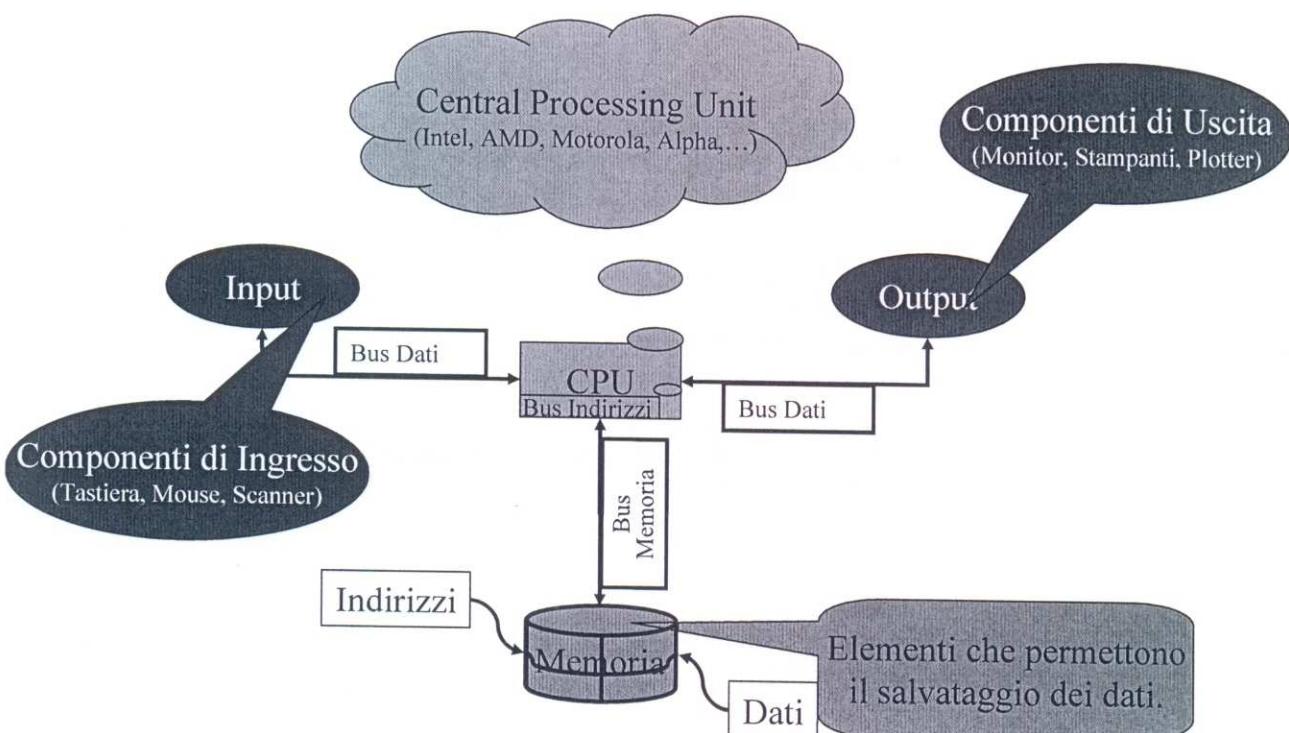
I Bus: strutture e meccanismi che provvedono alla comunicazione fra processori, memoria centrale, moduli di I/O.



Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

7

Moduli Funzionali di un Sistema di Elaborazione



Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

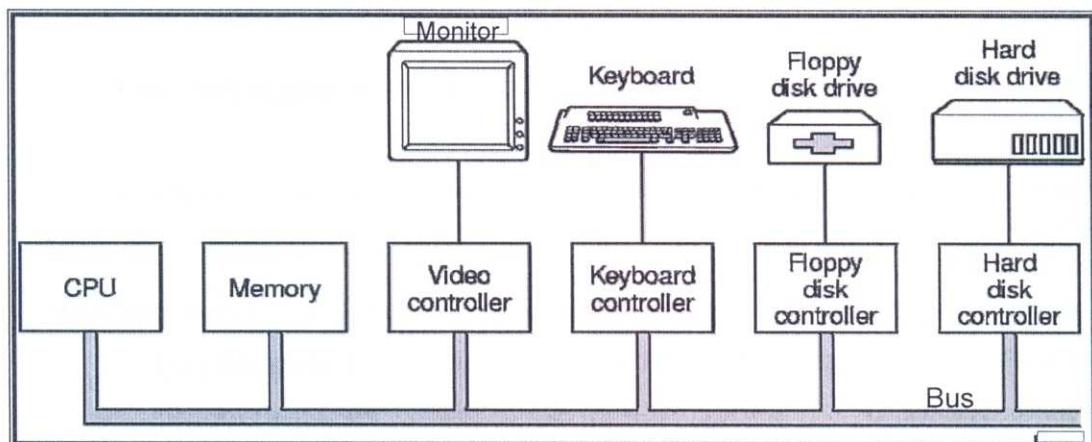
8

Frasi celebri (da una lezione di P. Ciancarini)

- (Popular Mechanics, 1949) Nel futuro i computer arriveranno a pesare non più di una tonnellata e mezzo.
- (Thomas Watson, presidente di IBM, 1943) Penso che ci sia mercato nel mondo per non più di cinque computer.
- (Editor di libri scientifici di Prentice Hall, 1947) Ho girato avanti e indietro questa nazione (USA) e ho parlato con la gente. Vi assicuro che questa moda dell'elaborazione automatica non vedrà l'anno prossimo

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

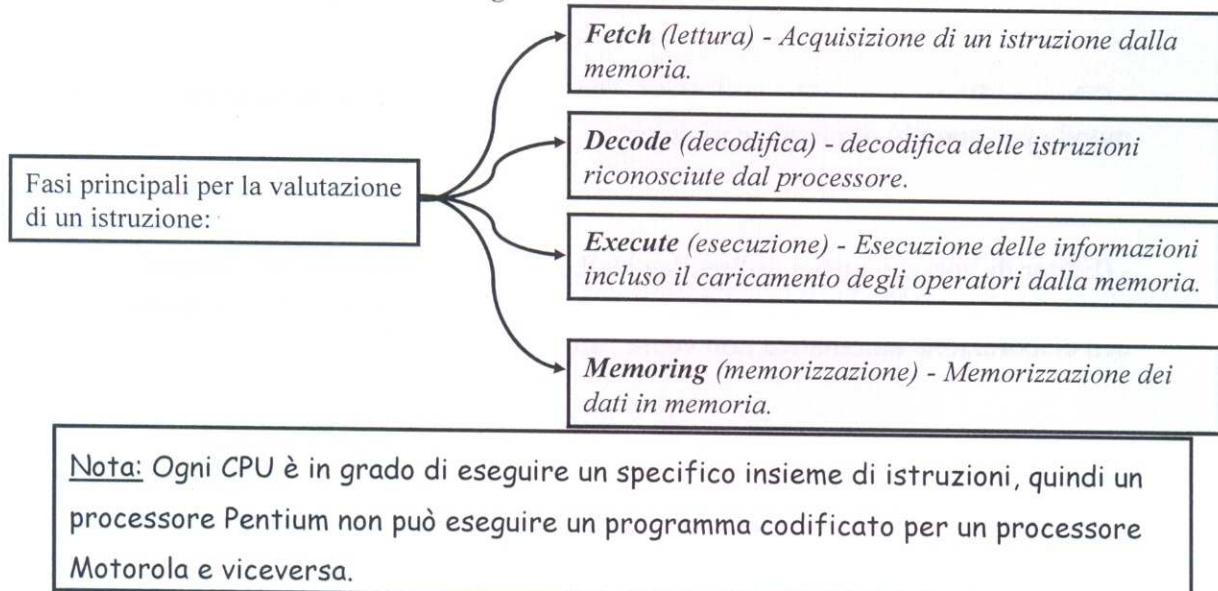
Componenti di un semplice Personal Computer



Il Processore

Il Processore è uno dei componenti esecutivi di un sistema di elaborazione, si occupa di recuperare le istruzioni dalla memoria e di eseguirle.

Il ciclo di azioni base eseguite da una CPU consiste nel *recupero* delle istruzioni dalla memoria, *decodificarle* per determinarne quale tipo di operatori sono presenti nell'istruzione e quali operandi sono necessari, ed infine *eseguirle*.



Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

11

Registri principali di un Processore

Program Counter (PC): contiene l'indirizzo di memoria della prossima istruzione da processare (Fetch).

Instruction register (IR): contiene l'ultima istruzione acquisita dalla memoria.

Memory Address Register (MAR): contiene l'indirizzo di memoria per la prossima operazione di lettura.

Memory Data Register (MDR): contiene i dati che devono essere letti o da scrivere in memoria.

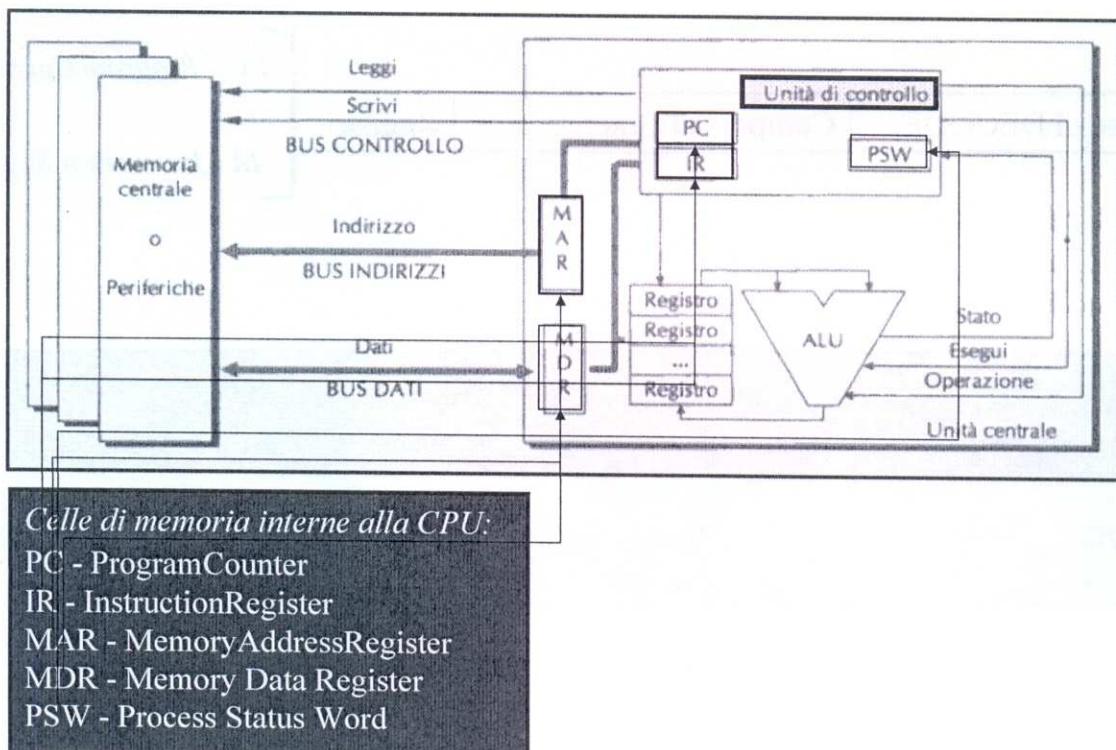
Program Status Word (PSW): contiene informazioni sullo stato di un processo, come il bit per l'abilitazione o la disabilitazione degli interrupt e il bit per la selezione del modo supervisore/utente.

Stack Pointer (SP): contiene l'indirizzo alla cima dello stack corrente di memoria. Tale stack è una struttura (frame) ed è generalmente utilizzata per conservare procedure iniziate ma non terminate, ogni frame contiene i parametri di ingresso e di uscita della procedura, le variabili locali e temporanee che non vengono mantenute nei registri.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

12

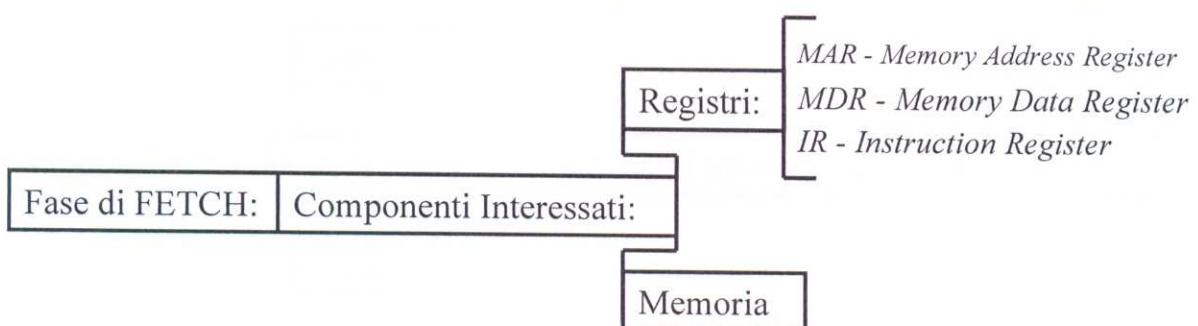
Registri principali di un Processore



Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

13

Descrizione delle fasi principali per l'esecuzione di un istruzione



Descrizione

F1) L'unità di controllo scrive l'indirizzo dell'istruzione da eseguire nel registro MAR, successivamente viene scritta ed inviata l'istruzione "LEGGI" sul BUS Unità-Centrale Memoria.

F2) La memoria seleziona la cella contenente l'istruzione e inserisce il contenuto nel registro MDR.

F3) L'unità centrale legge l'istruzione dal registro MDR e la memorizza nel registro IR.

Fase di DECODE:	Componenti Interessati:	Registri:
-----------------	-------------------------	-----------

PC - Program Counter

IR - Instruction Register

Descrizione

- D1)** L'unità di controllo incrementa il contenuto del registro PC affinché esso identifichi la successiva istruzione da eseguire.
- D2)** L'unità di controllo esamina l'istruzione presente nel registro IR e determina le operazioni da svolgere.

Fase di EXECUTE:	Componenti Interessati:
------------------	-------------------------

Registri

Memoria

Device

Descrizione

- E1)** Le unità interessate (Device) vengono opportunamente comandate, prelevando eventuali operandi dalla memoria e trasferendoli negli opportuni registri.

La Memoria

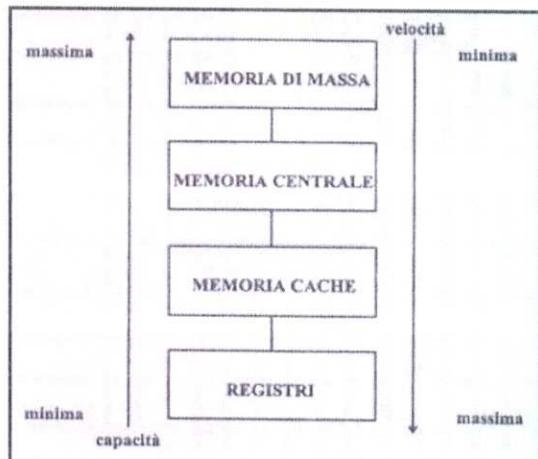
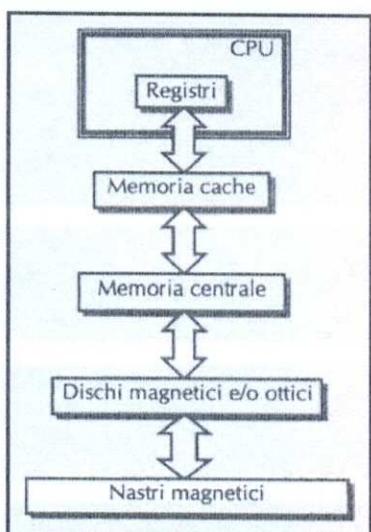
La memoria centrale è la seconda componente essenziale di un elaboratore, essa rappresenta l'unica area di memoria di grandi dimensione direttamente accessibile dalla CPU. Essa è strutturata come un vettore di parole o byte di dimensione variabili. Ciascuna parola possiede un proprio indirizzo. L'interazione tra CPU e Memoria avviene per mezzo di una sequenza di istruzioni **Load** e **Store** opportunamente indirizzate.

Problematiche:

- Identificazione delle parti di memoria libere e/o occupate.
- Distribuzione della memoria ai processi.
- Memoria Reale/Virtuale

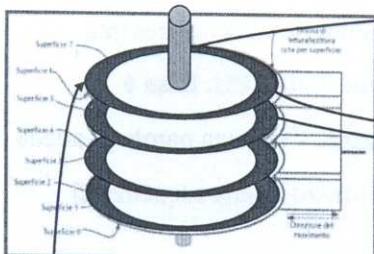
La memoria ideale dovrebbe essere **estremamente veloce** (più veloce dell'esecuzione di un'istruzione, in modo che la CPU non sia rallentata dalla memoria) **abbondante e a bassissimo costo**

La Memoria

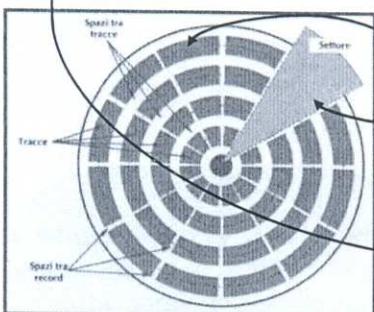


ESEMPIO DI MEMORIA DI MASSA

Dischi Magnetici



- Un disco magnetico è costituito da un insieme di *piatti* con due superfici che ruotano attorno ad un *perno centrale* con una ~~velocità di rotazione pari a 5400, 7200 o 10800 rpm.~~
- Ogni superficie è dotata una *testina* di lettura/scrittura.



- Le superfici sono organizzate in cerchi concentrici detti *tracce* e in spicchi di uguale grandezza detti *settori*.
- Tutte le tracce equidistanti dal centro, poste su piatti diversi, formano una superficie detta cilindro.
- I dati sono scritti occupando posizioni successive lungo le tracce.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

19

SULLA MEMORIA DI MASSA



Definiamo memoria di massa un componente informatico in grado di contenere tutti i programmi che possono servire all'utente insieme ai propri dati ed inoltre essere in grado di mantenere i dati per un tempo indefinito anche in assenza di elettricità. Il tempo di accesso alle memorie di massa è quasi tre ordini di grandezza più lungo delle memorie centrali.

I principali dispositivi per realizzare le memorie di massa sono basati su tecnologia magnetica e ottica.

La Memoria di massa può avere capacità che varia da pochi MEGAbyte a centinaia di GIGAbyte.

Il suo accesso può essere sequenziale (nastri magnetici) o ad accesso diretto (dischi magnetici o ottici).

Alcune Memorie Particolari

ROM

ReadOnlyMemory, la sua natura statica la rende inalterabile ed è quindi denominata *memoria non volatile*. Generalmente il suo contenuto non è modificabile dall'utente ed è programmato dalla casa produttrice.

EEPROM

Electrically Erasable Programmable Rom, anch'esse memorie *non volatili* come le ROM ma al contrario di esse possono essere cancellate e riscritte.

Memoria *volatile* è utilizzata da molti elaboratori per memorizzare ora e data corrente, ed altre informazioni essenziali per la configurazione dell'elaboratore.

CMOS

La memoria CMOS ed il circuito che in essa incrementa il tempo sono alimentati da una piccola batteria o accumulatore in modo che l'ora sia aggiornata correttamente anche quando l'elaboratore è scollegato dalla rete elettrica.

Quando la batteria si scarica l'elaboratore assume comportamenti simili a crisi epilettiche.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

21

Esempio d'uso della memoria per contenere due o più programmi

Per tenere due o più programmi contemporaneamente nella memoria centrale occorre risolvere due problemi:

- Come proteggere i programmi l'uno dall'altro, e il kernel del SO dai programmi.
- Come gestire la rilocazione.

Il problema della rilocazione è fortemente sentito in quanto risolve il problema dell'indirizzamento del codice e dei dati.

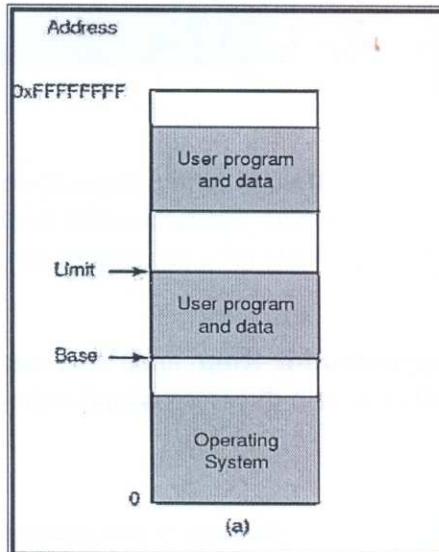
Esempio:

Supponiamo di avere un programma compilato e linkato con indirizzo della prima istruzione pari a zero, l'istruzione in posizione zero fa riferimento ad un indirizzo in posizione 10000, supponiamo inoltre che tutto il programma è i dati siano caricati a partire dalla posizione 50000; ne segue che quando il programma sarà posto in run, esso andrà in errore poiché farà riferimento a un indirizzo non appartenente al programma.

La soluzione a tale problema viene data dalla tecnica della rilocazione e dall'uso di due registri: **registro Base** e **registro Limite**.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

22



Esempio: un solo programma è caricato nella memoria centrale

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

23

Controllo e Corrispondenza

Definiamo Controllo e Corrispondenza la conversione dell'indirizzo generato dal programma, chiamato *indirizzo virtuale*, all'indirizzo della memoria, chiamato *indirizzo fisico*.

Definiamo MMU (Memory Management Unit) il dispositivo che effettua il controllo e la corrispondenza degli indirizzi.

Considerazione: Considerato che i registri contengono sia i riferimenti ad un programma sia i dati ad esso relativi, ogni qualvolta che si deve caricare un programma in memoria è necessario modificare tali registri e questo implica un dispendio di tempo.

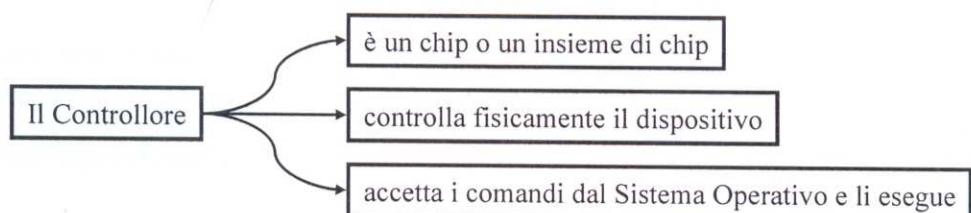
Quindi la riassegnare della CPU da un processo ad un altro è un'azione estremamente costosa.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

24

Dispositivi di I/O

I dispositivo di I/O generalmente sono composti da due parti: un **Controllore** e il **Dispositivo stesso**.



Esempio: controllore di un disco.

Il controllore di un disco può ricevere il comando di leggere l'informazione posta nell'indirizzo 13450, esso deve convertire tale numero in una posizione di memoria fisica identificata da: cilindro, settore e testina. Un ulteriore complicazione deriva dal fatto che i cilindri esterni hanno più settori di quelli interni, inoltre è possibile trovare settori o cilindri danneggiati.

Complicazioni dal punto di vista meccanico: si deve ruotare il disco e muovere il braccetto delle testine dopo averlo individuato, raggiunta la giusta posizione leggere i bit presenti sulla superficie magnetica e assemblarli in parole, effettuare un controllo d'errore ed infine inviarli al richiedente.

Per svolgere tali compiti spesso i controllori contengono piccoli elaboratori embedded

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

25

- Quindi ogni controllore di dispositivo è diverso da un altro, ognuno di essi ha un proprio software per la gestione e va caricato nel Sistema Operativo.
- Il software che gestisce il controllore del dispositivo di I/O è comunemente chiamato Device Driver (DRIVER), è fornito dal costruttore del dispositivo ed è Dipendente da S.O. su cui si vuole installare il Dispositivo.
- In generale un Driver per essere usato deve essere caricato in modalità Kernel;
Esistono tre modi per caricare un driver nel kernel:
 - 1) **Effettuare nuovamente** i link del kernel e riavviare in Sistema.
 - 2) **Effettuare la registrazione** del Driver in un file di sistema dicendo che vi è necessità di un driver, quindi riavviare il sistema. Al momento del boot il sistema cercherà i driver e li carica.
 - 3) **Effettuare il caricamento dei Driver** durante l'esecuzione e installarli al volo senza riavviare il Sistema. (**Driver dinamici** come USB o IEEE1394 detto anche FireWire).

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

26

Attività di Ingresso e Uscita

Le strategie per le attività di ingresso uscita possono riassumersi in tre modi diversi:

Busy Waiting	Interrupt	DMA
--------------	-----------	-----

Busy Waiting

- 1) Un programma Utente emette una chiamata di sistema per un dispositivo
- 2) Il sistema (kernel) la traduce in una chiamata alla procedura del driver appropriato
- 3) Il Driver darà il via all'operazione e restituisce, alla fine dell'operazione, i dati al kernel.
- 4) Il Kernel restituisce il controllo al programma chiamante.

Difetto: ha lo svantaggio di tenere occupata la CPU fin quando l'operazione non è terminata.

Interrupt

- 1) Il driver fa partire il dispositivo
- 2) Il S.O. blocca il programma chiamante
- 3) Quando il dispositivo ha finito il suo compito, manda un Interrupt al driver
- 4) Quando il controllore rileva la fine di un trasferimento dati lo segnala al S.O. attraverso un Interrupt.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

27

Attività di Ingresso e Uscita

DMA

- 1) Uso di un Chip speciale per le operazioni di Input/Output.
- 2) Controlla il flusso dei dati tra le memorie e i controllori di dispositivi senza l'uso della CPU.
- 3) La CPU avvia il DMA specificando: quanti bit devono essere trasferiti, il dispositivo interessato, gli indirizzi di memoria coinvolti, ect.
- 4) Al termine dell'esecuzione il DMA genera un Interrupt per catturare l'attenzione della CPU.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

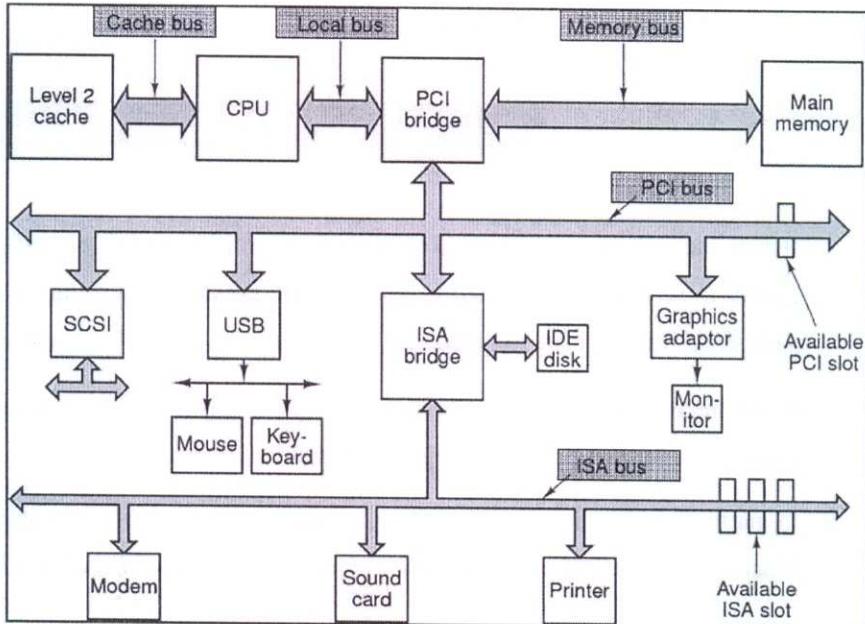
28

I BUS

Definizione: Un **BUS** può essere definito come un insieme di “fili”, è supportato da un protocollo di trasmissione rigorosamente definito che specifica l’insieme dei messaggi che si possono inviare attraverso “fili”.

In termini elettronici, i MESSAGGI sono inviati tramite configurazioni di livelli di tensione elettrica applicati ai fili con una predeterminata scansione temporale.

Sono presenti almeno 8 BUS:
Cache, Memoria, Locale, PCI, SCSI, USB, IDE, ISA.



Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

29

Caratteristiche di alcuni BUS

ISA: Industry Standard Architettura, proposto da IBM per i PC/AT lavora a 8,33 MHz e può trasferire 2 byte per volta, con una velocità massima di 16,67Mb/sec, è ancora presente su alcune schede madri per mantenere la compatibilità con le vecchie schede di I/O.

PCI: Peripheral Component Interconnect, inventato da INTEL come successore di ISA, lavora a 66 MHz, e trasferisce 8 byte per volta per una velocità massima di 528MB/sec

IDE: Il bus IDE serve per il collegamento di periferiche come dischi e CD-ROM al Sistema.

SCSI: Small Computer System Interface, è un bus a alte prestazioni, destinato a dischi veloci, scanner e altri dispositivi che necessitano di una considerevole larghezza di banda, può lavorare fino a 160Mb/sec.

USB: Universal Serial Bus, inventato per collegare al calcolatore dispositivi di ingresso/uscita lenti, ma esistono versioni di dischi, lettori CD-ROM e masterizzatori per questo BUS, in generale dispositivi esterni. USB è un bus centralizzato nel quale, un dispositivo radice interroga tutti i dispositivi presenti di questo tipo ogni millisecondo, esso può gestire un carico complessivo di 1,5MB/sec. Tutti i dispositivi USB condividono lo stesso drive evitando di conseguenza di dover installare un driver per ciascun nuovo dispositivo.

IEEE 1394: molto spesso chiamato FireWire (versione che Apple ha fatto del IEEE 1394), anch'esso seriale come l'USB ma progettato per trasferimento a pacchetto alla velocità massima di 50MB/sec. Collegamenti possibili sono le telecamere digitali e altri sistemi multimediali.

Introduzione ai Sistemi Operativi

Definire esattamente un Sistema Operativo è abbastanza difficile, ciò è dovuto al fatto che il S.O. realizza due funzionalità che sono praticamente scorrelate:

- Sistema Operativo come macchina estesa;
- Sistema Operativo come gestore delle risorse.

Sistema Operativo come Macchina Estesa

Il sistema operativo nasconde al programmatore la verità sull'hardware (*immaginate di dover gestire la scrittura o la lettura su floppy*), esso presenta una semplice interfaccia con cui lavorare per astrazione (es. i file non sono bit da leggere o scrivere, bensì nomi da aprire o chiudere).

Il sistema operativo cela tutta una serie di spiacevoli azioni che riguardano: l'interruzione, i timer, la gestione delle memorie ed altre problematiche di basso livello.

In questo senso la funzionalità del S.O. è quella di presentare all'utente una **Macchina Estesa** o Macchina Virtuale che sia più facile da programmare rispetto all'hardware sottostante.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

31

Sistema Operativo come Gestione delle Risorse

Il sistema operativo, visto come entità che ha come compito principale quello di mettere a disposizione dell'utente una comoda interfaccia per controllare e gestire tutte le componenti di un sistema complesso, assegna le competenze per la gestione dell'allocazione delle risorse a programmi specifici.

Esempio: Tre programmi che vogliono usare una stessa stampante senza utilizzo del gestore delle risorse.

La gestione delle risorse comporta la loro condivisione sotto due aspetti: **rispetto al tempo** e **rispetto allo spazio**.

Rispetto al tempo: programmi e utenti diversi fanno a turno per usarla, “*si alternano*”.
Esempio: assegnazione del processore a più programmi

Rispetto allo spazio: ad ogni programma sarà assegnata una “*parte della risorsa*”.
Esempio: assegnazione della memoria ai programmi.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

32

Breve Storia dei Sistemi Operativi

I Sistemi operativi si sono evoluti durante il corso degli anni e poiché storicamente l'evoluzione di un sistema operativo è dipendente della macchina su cui era installato, è necessario affrontarlo in maniera parallela.

Prima Generazione (1945-55): Dopo gli inutili sforzi della fine dell'800 dovuti a Babbage per la realizzazione della Macchina Analitica, si dovette aspettare sino alla metà degli anni 40 per riuscire a realizzare macchine da calcolo.

H. Aiken (Harvard)

John Von Neumann (Institute for Advanced Study di Princeton)

Konrad Zuse (Germania)

Queste macchine erano enormi, riempivano intere stanze ed erano milioni di volte più lente del più economico dei PC attuali. Non esisteva un linguaggio di programmazione così come lo conosciamo adesso, e la programmazione avveniva attraverso delle schede a spinotto, l'elaborazione più semplice durava ore a meno che non si bruciasse qualche valvola.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

33

Seconda Generazione (1955-65): L'introduzione dei transistor durante la metà degli anni '50 cambiò radicalmente l'impatto degli elaboratori con la società. I calcolatori divennero abbastanza affidabili da poter essere costruiti e venduti, tali calcolatori presero il nome MAINFRAME.

Dove sistemarono i MAINFRAME: erano tenuti in grosse sale macchine e al loro funzionamento erano assegnati gruppi di operatori specializzati.

Costo di un MAINFRAME: il loro costo si aggirava intorno a svariati milioni di dollari e poteva essere sostenuto da grosse compagnie, agenzie governative o da università.

Come si utilizzava un MAINFRAME: per fare girare un Job il programmatore doveva prima scrivere il programma su carta, successivamente doveva essere trascritto su schede perforate ed infine portato dall'operatore preposto per il suo caricamento sull'elaboratore.

Per migliorare i tempi di elaborazione, si adottò la strategia dei **Sistemi Batch**: i Job venivano caricati in dei nastri magnetici, con degli elaboratori meno costosi e specializzati, questi venivano dati in pasto all'elaboratore per la produzione dei dati di output che venivano caricati su di un altro nastro.

Si cominciano ad intravedere i primi sistemi operativi tra i quali il FMS (Fortran Monitor System) e IBSYS per l'IBM 7094

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

34

Terza Generazione (1965-80): In tale periodo le grandi case costruttrici avevano due linee di produzione, una orientata ai grossi elaboratori e l'altra alle piccole macchine, tale dualità rendeva costosa la produzione delle stesse e quindi l'IBM tentò di risolvere il problema.

Realizzò un insieme di macchine tutte simili tra di loro (SYSTEM 360), dove le caratteristiche di sistema operativo e di interoperabilità erano rispettate, in teoria lo stesso programma funzionava sia sulle macchine piccole sia su quelle con grosse prestazioni.

Vizi e virtù del sistema operativo orientato alle famiglie di macchine:

La virtù di tale sistema è facilmente individuabile, la grande forza nell'elaboratore è la versatilità, di contro l'intenzione di avere un software che girasse su tutti i modelli con caratteristiche estremamente differenti, modalità di calcolo orientate sia al calcolo scientifico sia a quello commerciale, gestire indifferentemente poche o molte periferiche, rese la sfida difficoltosa.

Vizi: l'IBM produsse un S.O. estremamente complesso in cui difficilmente si individuavano gli errori e quando venivano individuati la loro correzione portava ad altri errori, tanto da mantenere il numero degli errori costante nel tempo.

Alcune altre caratteristiche vennero fuori durante quegli anni:

Il MultiTasking, Lo Spooling, Il Timesharing,

Sistemi operativi come il MULTICS, antesignano dello UNIX, nacquero in tale periodo.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

35

Quarta Generazione (1980-oggi): lo sviluppo dei circuiti a larga scala (LSI - Large Scale Integration), chip con migliaia di transistor, permise la realizzazione di microprocessori e quindi dava ad ogni singolo individuo la possibilità di avere il proprio elaboratore (personal computer).

INTEL ebbe un ruolo predominante per la divulgazione dei personal computer, il suo processore l'8080 (1974 - CPU a 8bit) necessitava di un sistema operativo per il collaudo.

KILDALL, un consulente della Intel, scrisse il primo sistema operativo (CP/M - Control Program for Microcomputer) che permetteva di controllare il processore attraverso un disco floppy. Dalla realizzazione del CP/M e dalla diffidenza di INTEL nella produzione di microprocessori basati su disco, KILDALL fondò la DIGITAL RESEARCH (1977) per la produzione e vendita del CP/M per tutta una serie di processori tra cui l'8080, Zilog 80.

IBM nei primi degli anni '80 progettò il suo primo PC/IBM e cercando un software compatibile con il processore contattò Bill Gates per cedere la licenza del suo software (BASIC) e come consulente fu incaricato di ricercare un sistema operativo che andasse bene per il loro PC/IBM, ovviamente Gates suggerì il sistema di KILDALL (CP/M) come sistema operativo, ma KILDALL contattato da IBM rifiutò la proposta IBM.

IBM ritornò dal suo consulente (GATES) per una nuova proposta e a questo punto "il consulente" dopo avere acquisito un sistema operativo (DOS - Disk Operating System) fatto da una ditta locale la Seattle Computer Products, propose a IBM il sistema Operativo MS-DOS della neo azienda Microsoft contenente il DOS della Seattle Computer Products e il suo Basic. Da allora molte versioni si sono succedute a MS-DOS (Win3.xx, Win95, Win98, WinXX) e ancora oggi il mondo dei PC è gestito da sistemi operativi prodotti da Microsoft.

Altre svolte salienti di tale generazione sono la creazione di interfacce utente grafiche (GUI) da parte di DOUG ENGELBART dello Stanford Research Institute a cui sia il mondo **APPLE** sia Windows si sono inspirati per la realizzazione delle più moderne interfacce utenti per sistemi operativi orientati a utenti non esperti.

Inoltre si sono ispirate le produzioni di sistemi operativi alternativi quali UNIX con tutti i suoi derivati LINUX in testa, e i Sistemi Operativi Distribuiti basati su architetture di rete.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

37

Breve Classificazione dei Sistemi Operativi

Sistemi operativi per i mainframe

I sistemi operativi per i MainFrame sono fortemente orientati all'elaborazione di molti Job per volta, in cui ogni job necessita di una quantità enorme di dati di Ingresso/Uscita.

Tre tipi di servizi sono offerti: BATCH, TRANSAZIONALE, CONDIVISIONE DI TEMPO.

Sistemi operativi per i server

Uno scalino più in su' troviamo i S.O. per i sever (PC estesi, Postazioni di lavoro, micro mainframe), essi servono a più utenti in rete permettendo loro di condividere le risorse hardware, offrono servizi di stampa, servizi relativi ai file o al web. (ES. UNIX, WindowsNT/2000,Linux)

Sistemi operativi multiprocessore

Sono S.O. ottenuti come varianti dei S.O. per i server, hanno come caratteristica principale la comunicazione e la connessione. Gli elaboratori a cui si rivolgono sono chiamati Calcolatori Parallelisi, multicalcolatori, o multiprocessori.

Sistemi operativi per i personal computer

Prerogativa di tali S.O. è quella di fornire una buona interfaccia al singolo utente, sono ampiamente usati per l'elaborazione di testi, uso di fogli di calcolo e accesso a internet. Esempi tipici sono Windows XX, MACOS, linux, Leopard.

Sistemi operativi real-time

Caratteristica essenziale dei S.O. real-time è il tempo, le istruzioni sono accompagnate da un parametro “*tempo*” entro il quale esse devono essere espletate per garantire il buon funzionamento del sistema globale. Esempio nei processi di produzione industriale è importante che le macchine digitali facciano le operazioni in un certo intervallo di tempo per garantire il giusto funzionamento della catena di montaggio (catena di produzione di un’autostrada).

Sistemi operativi embedded

Tali S.O. sono estremamente particolari, nati per piccoli apparecchi come Palmari, Elettrodomestici e Telefoni cellulari devono essere caricati in ambienti con poca memoria e spesso orientati al real-time, esempi sono il PalmOs e Windows CE(Consumer Electronics), Simbian, Android.

Sistemi operativi per le smart card

I S.O. più piccoli vengono eseguiti sulle smart card (dispositivi della dimensione di una carta di credito), sono estremamente primitivi ed hanno un chip come CPU, stretti vincoli sull’alimentazione e sulle memorie, molti di essi sono orientati a una sola funzione. Altri hanno al loro interno un interprete per JVM (Java Virtual Machine) .

Concetti Base sui Sistemi Operativi

Concetti base o elementi comuni nei vari S.O. possono essere raggruppati SETTE grandi famiglie:

- 1. I processi**
- 2. Il deadlock**
- 3. La gestione della memoria**
- 4. L’ingresso/Uscita**
- 5. Il File System**
- 6. La Sicurezza**
- 7. La shell**

Introduzione ai Processi

Il Processo è un concetto chiave nell'ambito dei S.O., esso riduttivamente può essere pensato come un programma in esecuzione ... ma non è solo questo.

Cosa contiene o cosa è associato ad un PROCESSO:

Spazio di indirizzamento (address space, indirizzo iniziale e finale), il programma eseguibile, i dati del programma e il suo stack, un insieme di registri(PC, stack Pointer) e tutte le informazioni per l'esecuzione del programma.

NOTA: Un Processo ha al suo interno molte più informazioni del singolo programma da eseguire.

Un esempio per comprendere l'esigenza di salvare tali informazioni:

I sistemi time-sharing condividono tra più processi la CPU, quindi, quando un S.O. decide di bloccare un processo presente nella CPU esso, per essere ripreso in futuro, necessita di CONGELARE lo stato della CPU in quell'istante in modo da garantire la futura elaborazione dallo stato in cui si trovava. Tutte le informazioni necessarie in quell'istante devono essere conservate e riprese successivamente.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

41

La Tabella dei Processi

In molti sistemi operativi le informazioni di ciascun processo, tranne il contenuto del suo spazio di indirizzamento(programma eseguibile) sono memorizzate in una tabella di strutture del S.O., chiamata Tabella dei Processi (Process Table), una tabella per ogni processo esistente.

Da queste considerazioni si deduce che un processo può essere definito come un insieme di istruzioni, che rappresenta lo spazio di indirizzamento (Immagine in Memoria o Core Image), e dal corrispondente elemento nella Tabella dei Processi (Process Table Elements).

Processo
Core Image
Process Table Element

Un processo ha un sua vita e le principali chiamate al S.O. si occupano della CREAZIONE e TERMINAZIONE dei Processi.

Esempio: Al processo “shell di comando” viene richiesto di attivare un compilatore; esso, con una chiamata al sistema, produce un processo figlio per la compilazione e dovrà preoccuparsi della sua terminazione quando la compilazione è terminata

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

42

User ID

l'amministratore di sistema assegna un UID (User IDentification number) ad ogni persona autorizzata all'uso di un sistema, ogni processo in esecuzione ha l'UID della persona che lo ha lanciato, e ogni processo figlio ha lo stesso UID del processo che lo ha generato. Inoltre ogni utente fa parte di un gruppo di utenti ognuno dei quali ha un numero denominato GID (Group IDentification number)

Superuser

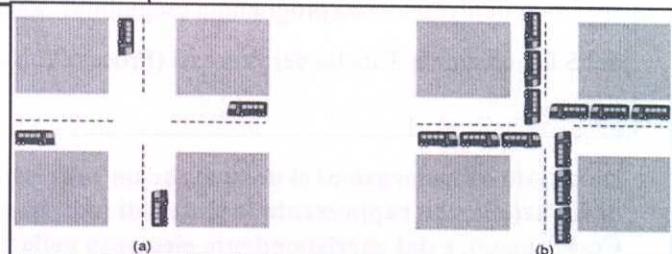
L'utente SuperUser per Unix o Administrator per WindowsXX ha dei poteri speciali, quindi può violare la maggior parte delle regole di protezione.

Introduzione ai Deadlock

DEADLOCK o situazione di stallo può avvenire quando due processi interagiscono tra di loro.

ESEMPIO 1:

A conseguenza della situazione in figura (a), può verificarsi la situazione in figura (b) in cui gli autobus arrivano insieme all'incrocio. Gli autobus restano bloccati (ingorgo incrociato) e nessuno dei capofila può fare retromarcia. La soluzione a tale problema non è semplice.



ESEMPIO 2:

Immaginiamo un elaboratore con un lettore di Nastri (N1) e un Masterizzatore (M1), inoltre immaginiamo due processi (P1,P2) che debbano utilizzare le due unità leggendo da N1 e scrivendo sul M1.

- 1) P1 richiede ed ottiene il Nastro;
- 2) P2 richiede ed ottiene l'uso del Masterizzatore;
- 3) P1 richiede il Masterizzatore e viene sospeso perché P2 non lo restituisce;
- 4) P2 richiede il Nastro ma anch'esso viene sospeso perché la risorsa è allocata da P1;

Introduzione alla Gestione della Memoria

Sistemi Operativi Elementari: In tali sistemi i programmi sono caricati in memoria UNO per volta e per caricare il secondo programma bisogna rimuovere il primo e successivamente caricare il secondo.

Sistemi Operativi Complessi: Permettono di mantenere contemporaneamente più programmi in memoria, e per evitare che interferiscano tra di loro e/o con il S.O. è necessario un meccanismo di protezione, presente nell'hardware e gestito dal S.O.

Gestione dello Spazio degli indirizzi: Un processo ha uno spazio di indirizzi che va da 0 a un certo valore di massimo. Nei casi più semplici lo spazio degli indirizzi è minore della memoria centrale (in questo caso il processo starà all'interno della memoria centrale e resterà spazio per ulteriori processi). Un caso più complesso è quando lo spazio degli indirizzi è maggiore degli indirizzi di memoria in questo caso si userà la tecnica della memoria VIRTUALE in cui soltanto una parte del processo (segmento) viene caricato sulla memoria centrale.

Nota: I molti calcolatori gli indirizzi sono a 32 o 64 bit e quindi consentono di indirizzare rispettivamente 2^{32} o 2^{64} byte.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

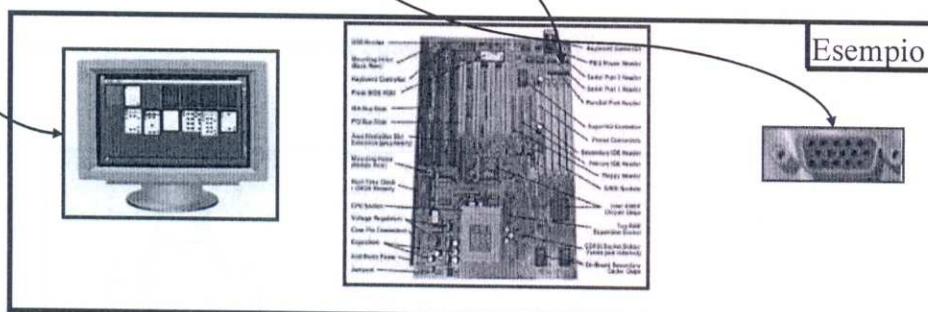
45

Introduzione all'I/O

Definizione: definiamo *Periferiche* quei componenti hardware la cui funzione principale è riferita alle comunicazioni tra il calcolatore e il mondo esterno.

Definizione: definiamo *Interfaccia* quei moduli funzionali che traducono i segnali interni al calcolatore in un formato comprensibile alle periferiche.

Definizione: definiamo *Porta* quel connettore predisposto sulla scheda di interfaccia accessibile dall'esterno in cui è collegata la periferica.



NOTA: Una parte del Software di I/O del S.O. è indipendente dal dispositivo è quindi può essere applicata a diversi dispositivi, un'altra parte (i DRIVER) sono strettamente legati al dispositivo e dipendono dalle specifiche del dispositivo stesso.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

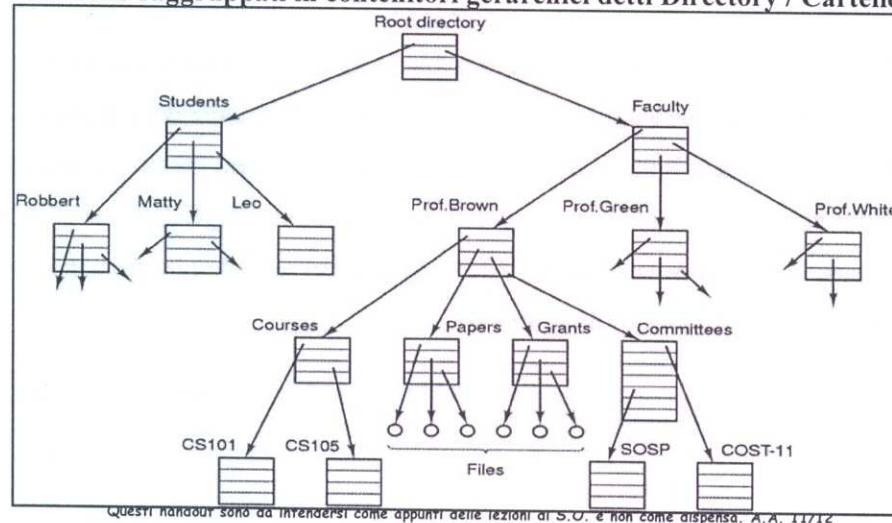
46

Introduzione ai File

Gestione dei file o File System, presente in tutti i sistemi operativi permette di fare **apparire all'utente i file indipendentemente dall'hardware che li contiene**. Così lo stesso file è rappresentato nello stesso modo sia che si trovi su floppy sia su disco sia su CD e sia che si trovi su di un unità a nastro o su di una memoria flash.

Ovviamente per far ciò è necessario l'intervento del S.O. e le chiamate che l'utente fa attraverso comandi testuali o iconici permettono al S.O. di localizzare il file per **scrivere, leggerlo o restituirne il suo contenuto all'utente**.

I file sono raggruppati in contenitori gerarchici detti Directory / Cartelle



47

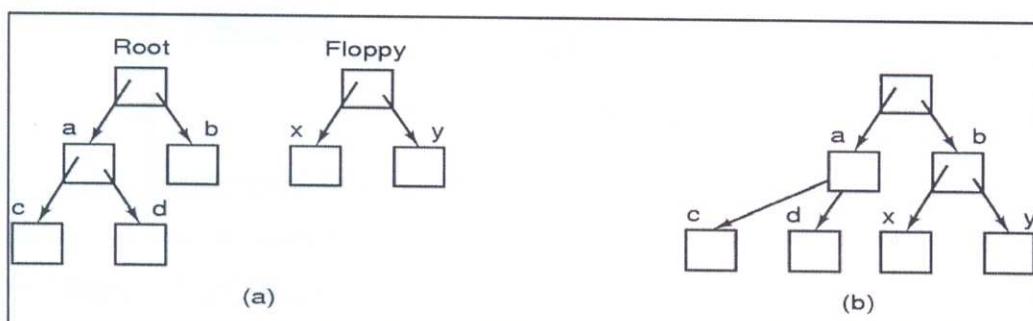
Azioni sui File

Il S.O. prima di accedere ad un file in lettura o in scrittura apre il file, ed è in questo momento che vengono controllati i **privilegi di accesso**. Se l'accesso è permesso ad ogni file viene assegnato un numero (descrittore del file) a cui il sistema farà riferimento per le operazioni future.

In UNIX, come in molti attuali S.O., il montaggio del file system relativo a un device è una attività estremamente importante.

Un floppy o un flash memory può essere considerato come una unità rimovibile pertanto il File System ad esso associato può essere montato (Mount) dinamicamente all'albero del file System gerarchico.

Ossia le gerarchie delle directory presenti nel DEVICE possono essere concatenate in qualsiasi parte dell'albero delle directory.



Azioni sui File [2]

I File Speciali:

I File Speciali sono forniti per fare in modo che i dispositivi di I/O sembrino file. In tale senso si può leggere o scrivere su di un dispositivo utilizzando le stesse chiamate di sistema previste per la lettura e scrittura di un file.

File Speciali a Blocchi, essi sono usati per fare I/O con dispositivi orientati ai blocchi, un esempio sono i dischi, l'istruzione di lettura del blocco N scavalca il File System interagendo direttamente con il dispositivo.

File Speciali a Carattere, stessa struttura di quella a blocchi ma sono usati, per fare I/O, con quei dispositivi il cui flusso di dati avviene con i caratteri, esempio tipico è la stampante.

Le PIPE:

La PIPE è un Pseudo-file utile per collegare due processi che devono scambiarsi dati. In tale senso i processi leggono o scrivono sulla pipe come se leggessero o scrivessero su e da un file, ma in realtà usano dei canali di comunicazione logici.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

49

Introduzione alla Sicurezza

La sicurezza è un altro aspetto importante nei sistemi di elaborazione, il S.O. ha al suo interno moduli che garantiscono riservatezza e sicurezza ai dati degli utenti.

Ad Esempio sistemi operativi come UNIX proteggono i file assegnando a ciascun file un codice binario a 9 bit (3 campi di 3 bit ciascuno), uno per il proprietario, uno per il gruppo a cui appartiene il proprietario e l'ultimo per tutti gli altri.

Ogni campo è costituito da 3 Flag (*rwx*), che se presente attiva la caratteristica sul file.

Esempio:

un file con codice di sicurezza *rwxr-x--x* autorizza la seguente sicurezza:

- I Campo) il **proprietario** può leggere, scrivere ed eseguire il file.
- II Campo) il **gruppo** a cui appartiene il proprietario del file può leggere e eseguire il file ma non può scrivere quel file(modificare).
- III Campo) gli **altri** possono eseguire il file ma non posso leggere o scrivere il file.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

50

Introduzione alla Shell di Comandi

La SHELL è anche l'interfaccia principale tra utente e S.O. a meno di interfacce grafiche.

La SHELL usa il terminale (tastiera, monitor) come ingresso/uscita standard, la sua visibilità avviene scrivendo il prompt dei comandi con cui possono essere impartiti comandi al S.O.

Alla scrittura di un comando il S.O. genera un processo figlio per l'esecuzione di tale comando e la shell resta in attesa di risposta.

Esempio

In UNIX esistono diverse SHELL:

sh, csh, ksh, bash

comandi particolari sono presenti nelle varie shell per mandare i processi in background e lasciare la shell libera per futuri utilizzi il più comune è il simbolo ‘&’ posto alla fine del comando.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

51

Le Chiamate di Sistema

Definiamo l'insieme di chiamate di sistema, fornite dal S.O., l'interfaccia tra il S.O. e i programmi utente.

Poiché i meccanismi per eseguire una chiamata di sistema sono fortemente dipendenti dalla macchina, il S.O. attraverso la sua interfaccia permette all'utente di non conoscere nulla della macchina sottostante.

Premesso che ogni calcolatore con una singola CPU può eseguire una singola istruzione alla volta. Se un processo che sta eseguendo un programma utente, in modalità utente, necessità di un servizio di sistema (e.i. lettura dati da un file), deve attivare una istruzione che esegua una chiamata di sistema per trasferire il controllo al sistema operativo.

Il S.O. riesce a capire cosa vuole il processo chiamante esaminando i parametri, successivamente esegue la chiamata di sistema e restituisce il controllo al processo chiamante ripartendo dall'istruzione successiva.

NOTA: Le chiamate di sistema possono essere considerate come delle chiamate a procedure utente con la variante che le chiamate di sistema interagiscono con il Kernel del S.O.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

52

Esempi di Chiamata di Sistema

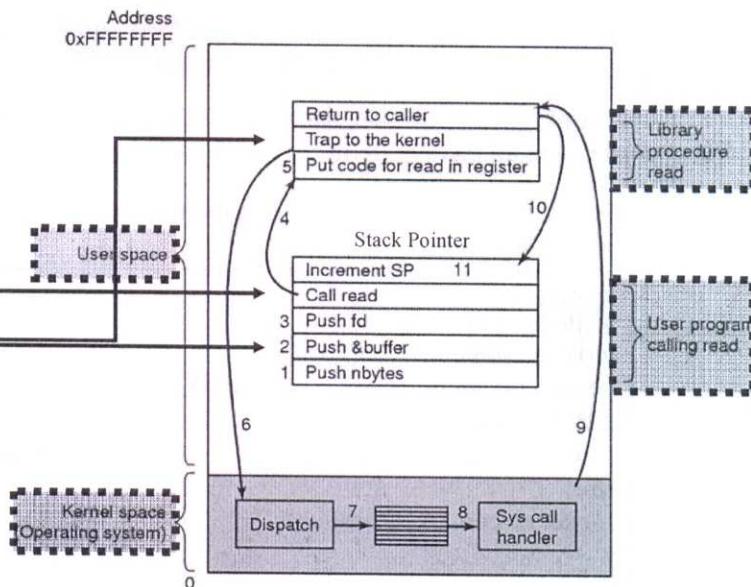
Esaminiamo il caso di una chiamata di sistema attraverso l'istruzione READ del linguaggio di programmazione C.

```
Cont = read(fd, buffer, nbytes);
```

Prima di richiamare la procedura di libreria *read*, che effettivamente esegue la chiamata *read* di sistema, il programma mette i parametri sullo stack. In seguito avviene la chiamata di libreria.

Successivamente mette in un registro il numero di chiamata della libreria ed infine esegue un TRAP per passare da modalità Utente a modalità Kernel.

•
•
•
•
•



Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

53

Esempi di Chiamata di Sistema

I servizi offerti dalle chiamate di sistema determinano quanto il sistema operativo deve fare per conto dell'utente, essi comprendono la creazione e la terminazione dei processi, creare, cancellare, leggere e scrivere file, gestione directory e Input/Output.

- Chiamate di sistema per la gestione di processi
- Chiamate di sistema per la gestione dei file
- Chiamate di sistema per la gestione delle directory

Chiamate di sistema per la gestione di processi

Le chiamate di sistema per la creazione di processi hanno i nomi più variegati, l'istruzione comune a diversi S.O. che realizza ciò è denominata FORK.

La chiamata di sistema FORK effettua la creazione di un processo come duplicato del processo originale (compresi tutti i duplicati).

In definitiva i due processi (padre e figlio) all'atto della FORK hanno gli stessi valori, ma successivamente le modifiche effettuate sull'uno NON vengono riportate sull'altro.

Inoltre al figlio viene assegnato un PID(Process Identification number) che permette la distinzione tra padre e figlio.

I processi figlio sono creati per permettere l'esecuzione di processi concorrenti diversi dai processi del padre.

```
#define true 1
while (true) {
    scrivi_prompt();
    Leggi_comando(comando,parametri);
    if FORK()!=0) {                                /* crea il processo figlio */
        waitpid(-1,&stato,0) }                      /* aspetta che il figlio termina */
    else
        execve(comando,parametri,0);             /* codice del figlio per eseguire il comando*/
    }
}
```

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

55

Chiamate di sistema per la gestione dei file

Per leggere o scrivere un file, esso, inizialmente deve essere aperto usando una OPEN di sistema.

Tale chiamata specifica quale file deve essere aperto e come deve essere aperto (creato, lettura scrittura o entrambi).

Il descrittore del file restituito dalla OPEN di sistema servirà in seguito per chiudere il file.

Altre istruzioni largamente usate sono READ e WRITE di sistema, queste permettono di scrivere o leggere un file, e anche se nella maggior parte delle volte i file sono sequenziali, a volte è necessario fare una lettura casuale del file.

L'istruzione di sistema SEEK permette di modificare il puntatore al file, permettendo la lettura ad accesso casuale del file.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

56

Chiamate di sistema per la gestione delle directory

Le chiamate di sistema a istruzioni per la gestione delle directory fanno riferimento al loro complesso piuttosto che al singolo file.

MKDIR, RMDIR, MOUNT sono alcuni esempi di istruzioni di sistema per la gestione delle directory.

Mount: Tale chiamata permette di fondere due File System in uno. Eseguendo la seguente chiamata di sistema:

```
MOUNT /dev/fd0 /mnt0
```

dove /dev/fd0 è uno speciale file a blocchi per il driver 0, /mnt è la posizione nell'albero in cui deve essere montato, o individua la modalità con cui deve essere montato il file System (lettura, scrittura, entrambi).

Dopo che la MOUNT è stata effettuata l'utente può navigare sul nuovo file system a partire dalla directory /mnt.

NOTA: L'istruzione MOUNT permette di integrare in un'unica struttura gerarchica F.S. riferiti a device eterogenei; dischi, floppy, flash ram, ecc

