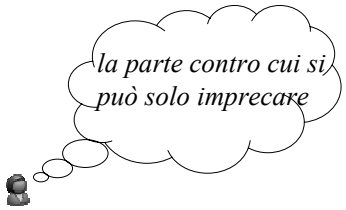
Il software



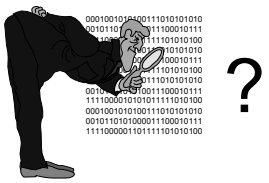
Il software

- L'hardware da solo non è sufficiente per il funzionamento dell'elaboratore ma è necessario introdurre il **software**
  - … ovvero un insieme di programmi che permettono di trasformare un insieme di circuiti elettronici in un oggetto in grado di svolgere delle funzioni di natura diversa

Il software

- Una programmazione diretta della macchina hardware da parte degli utenti è davvero difficile
  - l'utente dovrebbe conoscere l'**organizzazione fisica** dell'elaboratore e il suo linguaggio macchina
  - ogni programma dovrebbe essere scritto utilizzando delle **sequenze di bit** ed ogni piccola differenza hardware comporterebbe una riscrittura del programma stesso

Il software



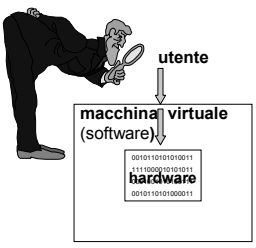
➢Questo non è accettabile ed è necessario fornire un meccanismo per **astrarre dall'organizzazione fisica della macchina**

Il software

- Inoltre, l'utente deve
  - ➢**usare nello stesso modo**, o comunque in modo molto simile, macchine diverse dal punto di vista hardware
  - ➢avere un **semplice linguaggio di interazione** con la macchina
  - ➢avere un insieme di programmi applicativi per **svolgere compiti diversi**

La macchina virtuale

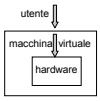
- Nei moderni sistemi di elaborazione questi obiettivi vengono raggiunti grazie alla definizione di **macchine virtuali** che vengono realizzate al di sopra della macchina hardware reale



La macchina virtuale

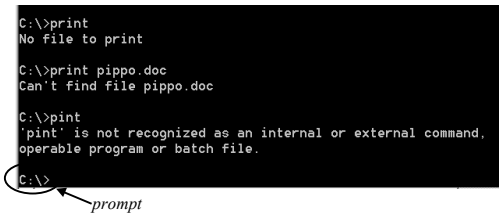
- Questa macchina si dice virtuale in quanto essa non esiste fisicamente ma viene realizzata mediante il software (**software di base**)
- L'utente interagisce con la macchina virtuale grazie ad un opportuno **linguaggio di comandi**

  La macchina virtuale si preoccupa della **traduzione** di ogni comando impartito dall'utente nella sequenza di comandi che realizzanno la stessa funzione e sono riconosciuti dalla macchina fisica sottostante



Linguaggio di comandi

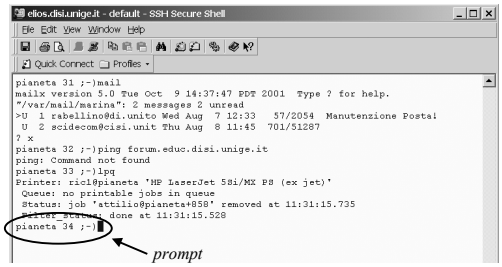
- Lavorando con una interfaccia a carattere i comandi vengono impartiti mediante la **tastiera**



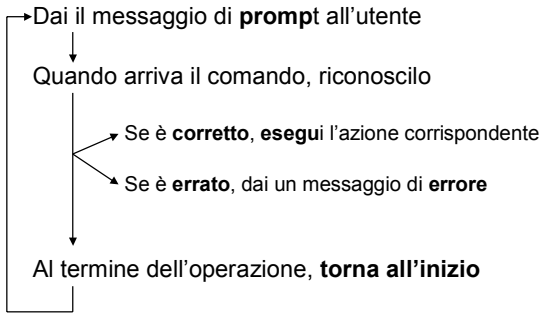
- Ogni comando ha un suo **nome** e una **sintassi** ben precisa

Linguaggio di comandi

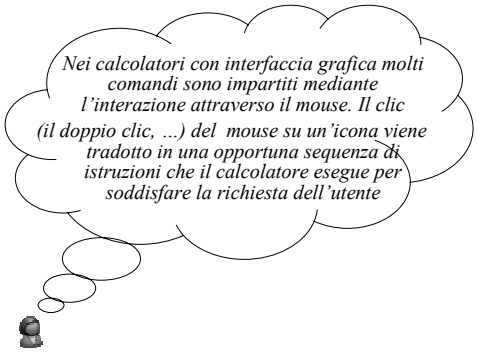
- In laboratorio vedremo i comandi che si usano con una **shell** implementata per Linux



Linguaggio di comandi: funzionamento della shell



Linguaggio di comandi



Software di base

- Gli strumenti software che permettono all'utente (e ai programmi applicativi) di gestire le risorse fisiche e di interagire con l'elaboratore in modo semplice sono parte della macchina virtuale
- Si parla di **software di base**, per denotare un insieme di programmi che, a livello macroscopico, offrono due classi di funzioni
  - funzioni proprie del **sistema operativo**
  - funzioni di **traduzione** tra linguaggi diversi

Il sistema operativo

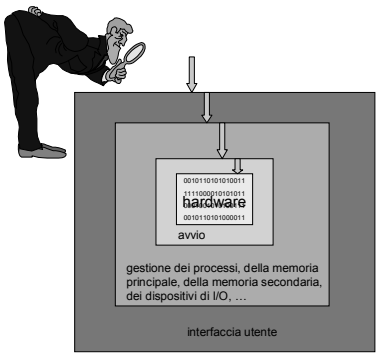
- È il componente **software fondamentale** di un sistema di calcolo
- È formato da un insieme di programmi che interagiscono tra loro per realizzare due obiettivi

1. Gestire efficientemente l'elaboratore e i suoi dispositivi
2. Creare un ambiente virtuale per l'interazione con l'utente

Funzioni principali del sistema operativo

- Avvio dell'elaboratore
- Gestione del processore e dei programmi in esecuzione (detti **process**i)
- Gestione della memoria principale
- Gestione della memoria virtuale
- Gestione della memoria secondaria
- Gestione dei dispositivi di input / output
- Interazione con l'utente

Il sistema operativo

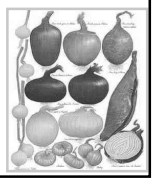


Il sistema operativo

Dal punto di vista strutturale il sistema operativo è formato da un insieme di livelli, che formano la cosiddetta

struttura a cipolla

Idealmente l'utente è ignaro di tutti i dettagli delle operazioni svolte dai livelli inferiori della gerarchia e conosce solo le operazioni del livello più alto



Il sistema operativo

mono utente	**multi utente**
mono programmato	**multi programmato**

Il sistema operativo

- **Mono-utente** o **multi-utente** (mono/multi-user)
  - si distingue tra elaboratori di tipo personale e elaboratori utilizzabili da più utenti contemporaneamente
- **Mono** o **multi-programmati** (mono/multi-tasking)
  - si distingue tra elaboratori in grado di eseguire un solo programma alla volta oppure più programmi *"contemporaneamente"*

Funzioni principali del sistema operativo

- Avvio dell'elaboratore
- Gestione del processore e dei processi
- Gestione della memoria principale
- Gestione della memoria virtuale
- Gestione della memoria secondaria
- Gestione dei dispositivi di input / output
- Interazione con l'utente

Avvio dell'elaboratore

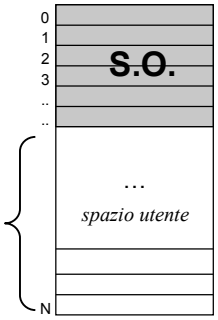
- Il sistema operativo viene mandato in esecuzione al momento dell'accensione del calcolatore
- Questa fase prende il nome di **bootstrap**
- In questa fase una parte del sistema operativo viene caricata nella memoria principale

Avvio dell'elaboratore

- In genere questa parte del sistema operativo comprende
  - i programmi per la gestione del processore
  - i programmi per la gestione della memoria
  - i programmi per la gestione dell'input/output
  - i programmi per la gestione delle risorse hardware
  - i programmi per la gestione del file system
  - un programma che crea l'interfaccia verso l'utente

Avvio dell'elaboratore

- Una parte del sistema operativo deve essere **sempre mantenuta in memoria principale** e deve essere sempre pronta per l'esecuzione



Avvio dell'elaboratore

- Spesso durante questa fase sono eseguiti anche dei programmi che verificano l'eventuale presenza di **virus** sul disco dell'elaboratore
- I virus sono dei programmi che possono essere trasmessi da un elaboratore ad un altro quando si copiano dei programmi oppure quando si salvano degli allegati dalla casella di posta elettronica
- Un virus può danneggiare il funzionamento dell'elaboratore, anche in modo piuttosto grave

Funzioni principali del sistema operativo

- Avvio dell'elaboratore
- Gestione del processore e dei processi
- Gestione della memoria principale
- Gestione della memoria virtuale
- Gestione della memoria secondaria
- Gestione dei dispositivi di input / output
- Interazione con l'utente

Esecuzione dei programmi

- Quando si scrive un comando (oppure si clicca sull'icona di un programma), il sistema operativo
  - cerca il programma corrispondente sulla memoria secondaria
  - copia il programma in memoria principale
  - imposta il registro Program Counter con l'indirizzo in memoria principale della prima istruzione del programma

Sistemi mono-utente, mono-programmati

- Un solo utente può eseguire un solo programma alla volta
- Il programma viene "lanciato", eseguito e quindi terminato
- Ma la CPU viene sfruttata al meglio?

Sistemi mono-utente, mono-programmati

- …. **no**, si spreca molto tempo!
- La CPU è molto più veloce dei supporti di memoria secondaria e delle altre periferiche, e **passa la maggior parte del suo tempo in attesa** del completamento delle operazioni demandate a questi **dispositivi**
- Durante l'attesa si dice che la CPU è in uno stato inattivo, detto **idle**

Esempio (1)

- Un processo è costituito da 1000 istruzioni e ogni istruzione richiede 1 microsec. per essere eseguita dalla CPU

  → tempo totale di esecuzione = $10^3 * 10^{-6}$ = 1 millisec.

- A metà esecuzione è richiesta la lettura di un dato dal disco. Il tempo di lettura è pari ad 1 millisec.

  → tempo totale di esecuzione = 2 millisec.

- Idle time = 1millisec.

  → corrisponde a **50%** del tempo totale di esecuzione ed è **tempo sprecato**

Esempio (2)

- Un processo è costituito da 1000 istruzioni e ogni istruzione richiede 1 microsec. per essere eseguita dalla CPU

  → tempo totale di esecuzione = 1 millisec.

- A metà esecuzione è richiesto un dato all'utente. Il tempo di reazione è pari ad 1 sec.

  → durata totale dell'esecuzione = 1001 millisec.

- Idle time = 1 sec.

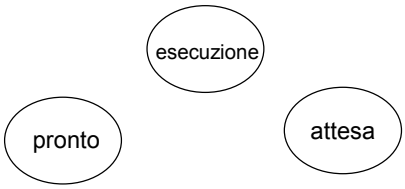
  → corrisponde al **99,9%** del tempo totale di esecuzione ed è **tempo sprecato** !!!!

Soluzione: sistemi multiprogrammati

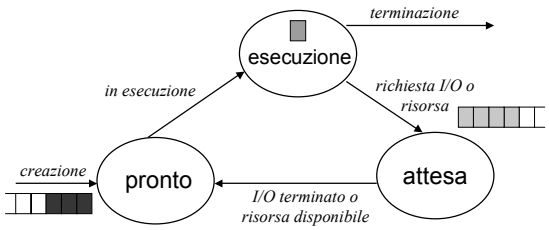
- Quando la CPU è nello stato di idle la si può sfruttare per eseguire (parte di) un altro processo
- Quando un processo si ferma (per esempio in attesa di un dato dall'utente) la CPU può passare ad eseguire le istruzioni di un altro processo
- Il sistema operativo si occupa dell'**alternanza** tra i processi in esecuzione

Soluzione: sistemi multiprogrammati

- Un processo può trovarsi in tre diversi stati: in **esecuzione**, in **attesa**, **pronto**



Soluzione: sistemi multiprogrammati



Soluzione: sistemi multiprogrammati

- Più programmi sembrano essere eseguiti "*contemporaneamente*"
- In realtà in esecuzione c'è sempre **un solo** processo ma, se l'alternanza è molto frequente, si ha un'idea di simultaneità
- Di solito è posto un limite al numero di processi "*contemporaneamente*" in esecuzione

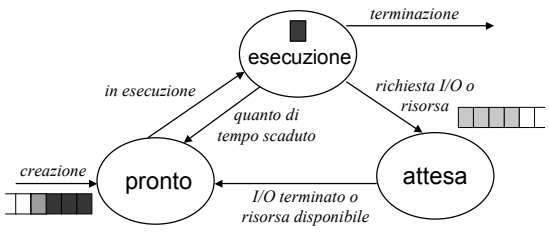
Cosa succede se …

- Un processo non si ferma mai in attesa di I/O o di una risorsa?
- Più utenti vogliono usare il computer?
- … è necessario far sì che la risorsa più importante del computer - la CPU - sia **distribuita** tra i processi dello stesso utente e di utenti diversi
- Si parla anche di **scheduling** del processore

Esempio di scheduling: Round Robin

- Ad ogni processo viene assegnato un **quanto** di tempo di CPU (**time slice**)
- Terminato il quanto di tempo, il processo viene sospeso e rimesso nella coda dei processi pronti (al fondo)
- La CPU viene assegnata ad un altro processo pronto
- Un processo può usare **meno** del quanto che gli spetta se deve eseguire operazioni di I/O oppure ha terminato la sua computazione

Esempio di scheduling: Round Robin

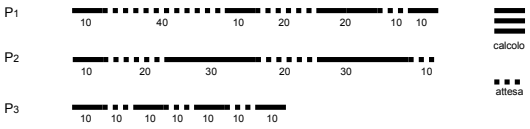


Effetti dell'alternanza tra processi

- L'esecuzione di più processi sembra avvenire realmente in parallelo
- Più utenti possono usare allo stesso tempo il computer, perché la CPU viene assegnata periodicamente (per esempio ogni 10 o 100 millisec.) ai processi dei vari utenti
- All'aumentare del numero di processi e del numero di utenti le **prestazioni** del sistema **possono degradare**

Esercizio

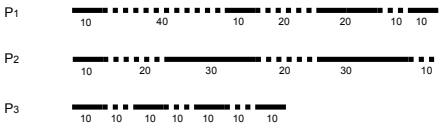
*Supponiamo di avere nella coda dei processi **pronti** tre processi P_1, P_2, e P_3 con i seguenti "comportamenti" in termini di computazione e tempi di attesa*



1. *Quante unità di tempo ci vogliono per portare a termine tutti e tre i processi in un sistema mono-programmato?*
2. *E in un sistema multi-programmato, se si applica l'alternanza tra i processi?*

Esercizio: soluzione

1. *Quante unità di tempo ci vogliono per portare a termine tutti e tre i processi in un sistema mono-programmato?*

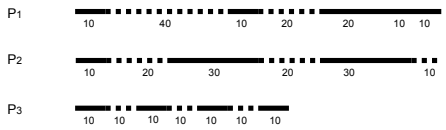


La soluzione qui è semplice, basta sommare i tempi totali dei tre processi
$tot(P_1)=120$
$tot(P_2)=120$
$tot(P_3)=70$

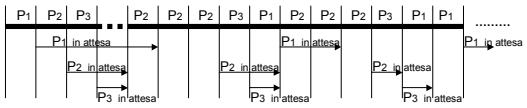
$tot(P_1+P_2+P_3)=120+120+70=310$

Esercizio: soluzione

2. *E in un sistema multi-programmato, se si applica l'alternanza tra i processi?*



*In questo caso, quando un processo va in attesa di un evento di I/O o di una risorsa, il processore viene assegnato al **primo** processo pronto*



Esercizio

*Supponiamo di avere nella coda dei processi **pronti** i processi*

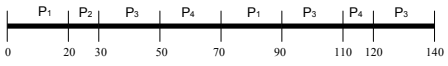
- *P_1 durata = 40 unità di tempo*
- *P_2 durata = 10 unità di tempo*
- *P_3 durata = 60 unità di tempo*
- *P_4 durata = 30 unità di tempo*

Qual è la sequenza di esecuzione con una politica di scheduling Round Robin e quanto di tempo pari a 20 unità?

Esercizio: soluzione

- *P_1 durata = 40 unità di tempo*
- *P_2 durata = 10 unità di tempo*
- *P_3 durata = 60 unità di tempo*
- *P_4 durata = 30 unità di tempo*

Bisogna mandare in esecuzione i processi pronti e interromperli (se non terminano prima) ogni volta che scade il quanto di tempo
(NB: non consideriamo eventuali tempi di attesa)



Altre politiche di scheduling

- **FIFO** (First Come First Served)
- **SJF** (Shortest Job First)
- **Priorità**

- **SRTF** (Shortest Remaining Time First)
  Versione preemptive della politica SJF

Guardate sul libro di testo per maggiori dettagli su queste politiche …

Gestione dei processi

- Per gestire un insieme di processi "*contemporaneamente*" attivi il S.O. mantiene, in una zona di memoria riservata, la **tabella dei processi** (**PCBT, P**rocess **C**ontrol **B**lock **T**able**)**
- Per ogni processo vi è un **descrittore** nel quale sono memorizzate molte informazioni, tra cui
  - L'identificatore del processo
  - L'identificatore dell'utente proprietario
  - Lo stato del processo
  - Il contenuto del registro Program Counter e degli altri registri
  - Informazioni sui file e sulle risorse in uso
  - Informazioni sull'utilizzo della memoria centrale e secondaria
  - Informazioni per lo scheduling

Gestione dei processi

- Tutte queste informazioni servono per realizzare l'operazione di **cambio di contesto** (**context switch**)
- Quando un processo rilascia la CPU, le informazioni sul suo stato vengono memorizzate nel suo descrittore all'interno della tabella dei processi
- In questo modo, quando tornerà nuovamente in esecuzione, il processo potrà **ripartire dal punto in cui era stato interrotto**

Gestione dei processi e della RAM

- Come abbiamo già detto più volte, i programmi per essere eseguiti devono essere caricati nella memoria principale
- È il sistema operativo che **coordina** le operazioni per la gestione dei processi e per la conseguente allocazione della memoria principale … ma ne parleremo in seguito