# Sistemi operativi

## Programmi e processi

- Programma: insieme statico di istruzioni
- Processo: entita' che tiene traccia dello stato dell'esecuzione di un programma
  - Posizione nel programma
  - Valori dei registri della CPU
  - Valori delle celle di M assegnate al programma
- Anche piu' processi per lo stesso programma
  - Es.: ho due documenti Word aperti contemporaneamente

## Programmi e processi

- Non si tratta di una distinzione formale...
- Un processo, infatti, può essere interrotto
- In tal caso è necessario "salvare" da qualche parte lo stato del processo, al momento dell'interruzione, in maniera da poter continuare la sua esecuzione dopo aver gestito l'evento che ha causato l'interruzione

## Esempio

- Ricetta = programma
- Noi = processore
- Ingredienti = dati in input
- Attivita' di leggere la ricetta, usare gli ingredienti, mescolare, cuocere = processo
- Telefonata durante l'esecuzione della ricetta = interrupt(segnale di interruzione: può indicare che si è verificato un errore o semplicemente che i dati richiesti sono finalmente disponibili…segnala un evento che deve essere gestito)

## Esempio:gestione interrupt

- Smettiamo di mescolare,… e prendiamo nota della riga della ricetta = il processo interrompe l'esecuzione, salva lo stato corrente e le informazioni di esecuzione
- Rispondiamo alla telefonata=gestiamo l'evento
- Alla fine della telefonata, l'esecuzione riprende dal punto in cui era arrivata e viene conclusa(il processo si dice terminato)
- Pranzo = output del processo

## Problema: non posso interrompere il processo in esecuzione

- …e se, nel frattempo, si brucia l'arrosto, si scuoce la pasta e ci và a fuoco la casa? Ovvero se non posso interrompere l'attività corrente?☹
- Semplicemente devo "ritardare" la gestione dell'interrupt = lasciamo squillare per un po' il telefono…☺
- Cerchiamo di terminare il più velocemente possibile le operazioni che stiamo effettuando e gestiamo l'interrupt(sperando che la persona in attesa sia molto paziente)

## Terminazione di un processo


Problema2: e se non si può far attendere la persona in linea(es. appuntamento ragazza\o)? Ovvero l'interrupt ha una priorità molto elevata e deve essere gestito immediatamente pena la perdita dell' interrupt o peggio? ☹

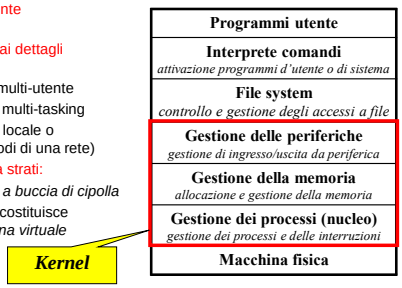
- …Niente pranzo oggi !!!= Dobbiamo "abortire"(abort) il processo ovvero terminarlo forzatamente(si usa anche il termine kill=uccidere) prima che il compito sia stato portato a termine(una specie di "Asta la vista baby" alla Terminator)☹☹☹
- …speriamo che almeno stasera si esca a cena!

## Che cos'è un Sistema Operativo?

- E' un insieme di programmi... dunque è un software
- E' necessario per utilizzare le "risorse" *hardware* del computer(memoria,periferiche,CPU,coprocessori, schede di espansione)...
- ...e quelle *software* (programmi)
- Il suo compito è quello di semplificare al massimo l'utilizzo di tali risorse fornendo un'interfaccia "user-friendly"(in italiano:amichevole per l'utente)
- ...garantendo contemporaneamente una gestione efficiente delle suddette risorse

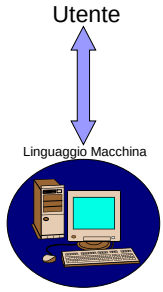
## Il Sistema Operativo

- E' uno strato software che opera direttamente sull'hardware
- Isola gli utenti dai dettagli dell'hardware
- Mono-utente o multi-utente
- Mono-tasking o multi-tasking
- Funzionamento locale o distribuito(sui nodi di una rete)
- E' organizzato a strati:
  - Architettura *a buccia di cipolla*
  - Ogni strato costituisce una *macchina virtuale*




## Macchine virtuali

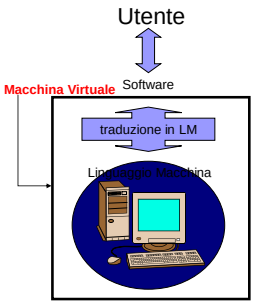
- Un computer è una macchina programmabile, tuttavia esso non è direttamente utilizzabile da parte degli utenti poiché richiederebbe la conoscenza dell'hardware della specifica macchina e del suo linguaggio macchina(associato al processore e al BIOS)
- Il linguaggio macchina è estremamente complicato e non di facile gestione




## Virtuale = Astratta

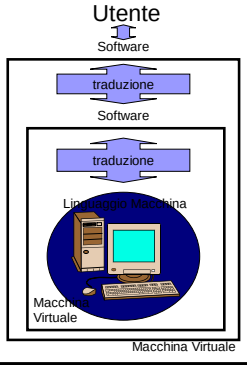
in contrapposizione a reale(fisica)

- In altre parole desideriamo astrarci dai dettagli fisici della macchina in oggetto e dal suo specifico linguaggio macchina
- L'idea è quella di realizzare al di sopra della macchina reale una macchina virtuale astratta che abbia le funzionalità desiderate e che sia facile da utilizzare per l'utente
- L'utente interagisce con la macchina virtuale, ogni comando viene poi tradotto nei corrispondenti comandi sulla macchina fisica
- La macchina virtuale è realizzata mediante software (programmi)




## Realizzare una macchina virtuale…

- La macchina virtuale viene realizzata mediante il software di base:
  - *Sistema Operativo*
  - *Linguaggi e ambienti di programmazione ad alto livello*: interpreti e compilatori
- Non vi sono limiti al numero e al tipo di macchine virtuali che possono essere realizzate
- In genere nelle macchine moderne sono strutturate su più livelli (*struttura a cipolla*)



## Moduli del sistema operativo

- Il *gestore dei processi* è responsabile dell'esecuzione dei programmi da parte dell'unità di elaborazione(CPU)
- Il *gestore della memoria* ha la funzione di allocare la memoria e partizionarla tra i vari programmi
- I *drivers* sono responsabili delle operazioni di *ingresso/uscita* che coinvolgono le periferiche
- Il *file manager* è responsabile della gestione dei file in memoria di massa
- *L'interprete dei comandi* consente all'utente di lanciare un programma, di esplorare il File System,etc… attraverso l'interfaccia testuale o grafica messa a disposizione dal SO

## A che serve il Sistema Operativo?

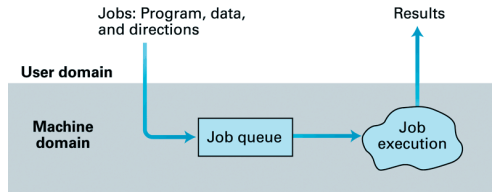
- Il sistema operativo, in ultima analisi, fornisce un **ambiente virtuale** con cui l'utente può interagire facilmente nascondedogli le complicazioni tecniche necessarie per realizzare le operazioni effettuate
- Fornisce ai programmi applicativi dei **comandi di alto livello** le *System Calls*(Invocazioni al sistema(operativo)) tramite i quali i programmi possono richiedere al SO di effettuare determinate operazioni su risorse(come i file) anch'esse virtuali!...del tipo: crea il file pippo.txt in una certa directory,scrivi nel file "hello world!",chiudi il file aperto etc...
- **Permette l'esecuzione di più programmi in parallelo**(nei sistemi **multi-tasking**) anche se in realtà in ogni istante un solo programma è in esecuzione sulla CPU(nelle macchine mono-processore)
- **Permette a più utenti di lavorare contemporaneamente**, nei sistemi multi-utente, **condividendo delle risorse software e hardware**(es. Stampante) e garantendo l' esclusività di altre
- **Permette a utenti diversi di utilizzare la stessa macchina come se vi lavorasse un solo utente...** Es:effettuando la login in WindowsXP, l'utente vede il desktop così come lo ha lasciato...
- **Permette di utilizzare più processori** per aumentare le prestazioni,
- **Permette di lavorare su memorie di massa diverse come se si trattasse dello stesso tipo di memoria e di effettuare operazioni simili su periferiche aventi hardware differenti** grazie ad appositi programmi (i drivers)

## Tutto ebbe inizio...

- Anni '40 e '50: un solo programma in esecuzione (job) alla volta:

  sistema mono-tasking
- Il computer è a disposizione del programma dall'inizio alla fine della sua esecuzione
- Coda dei job, gestita FIFO (first in, first out) e/o con priorita'
- L'utente inseriva una serie di schede perforate contenti il programma e i dati e aspettava(armato di una buona dose di pazienza e di caffè) che il programma terminasse e i risultati fossero disponibili
- L'apparecchiatura che leggeva in maniera automatica le schede perforate era detta *scheduler*

## Coda dei job



## Svantaggi dei sistemi monoprocessore mono-tasking

- Nessuna interazione utente-programma
- Lentezza: la CPU non puo' essere usata da nessun processo mentre il programma in esecuzione svolge operazioni di I/O
- MS-DOS(Microsoft Disk Operating System) e' un SO monotasking: non si puo' fare niente altro mentre si formatta un floppy o si memorizzano dati sul disco

## Time sharing

- Ripartizione del tempo di CPU tra tutti i processi che la vogliono
- Ogni job rimane in esecuzione solo per un quanto di tempo, poi l'esecuzione passa al prossimo job e il primo va in attesa
- Durata del quanto di tempo: tra 100 e 200 millisecondi
- A ciascun utente sembra di avere la CPU tutta per lui, l'unico effetto percepito è quello di un rallentamento del sistema(che dipende dal numero dei processi in esecuzione e dal numero di utenti nei sistemi multi-utente)
- Time –sharing in sistemi mono-processore: multi-tasking (piu' programmi in esecuzione con una sola CPU)

## Sistemi multiprocessore

- Reti di calcolatori: vari calcolatori che si scambiano dati
- Es.: Internet
- Una rete e' un sistema multiprocessore con una CPU su ogni calcolatore
- Anche singoli calcolatori con piu' CPU
- E' necessario non solo il coordinamento delle attivita' di ogni processore, ma anche bilanciamento del carico sui processori ovvero i compiti da eseguire (task) o loro parti(sub-tasks) devono essere suddivisi tra i vari processori

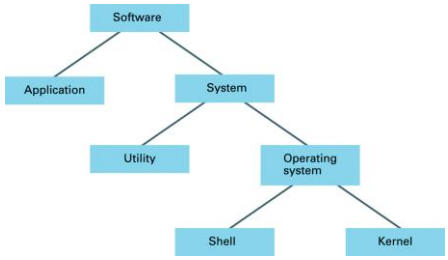
## Tipi di software

- Software applicativo: programmi per svolgere compiti particolari
  - Fogli elettronici
  - Programmi per la scrittura di testi(Text Editors,Word Processors)
  - Giochi ...
- Software di sistema: compiti comuni a tutti i calcolatori

## Software di sistema

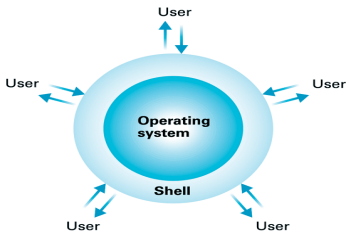
- Sistema operativo
- Software di utilita': aumenta le funzionalita' del SO
- Es.:
  - sw per comunicare via modem
  - sw per comprimere dati

## Suddivisione del software

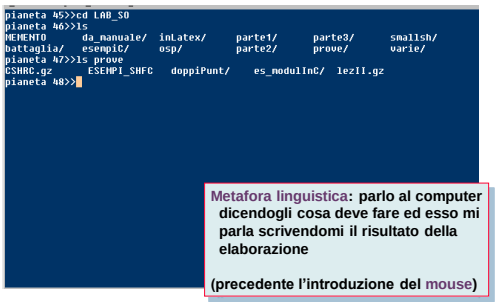



## Sistema operativo: shell

- Shell (guscio): interfaccia tra SO e utenti
- Di solito grafica (Graphic User Interface), ma anche testuale



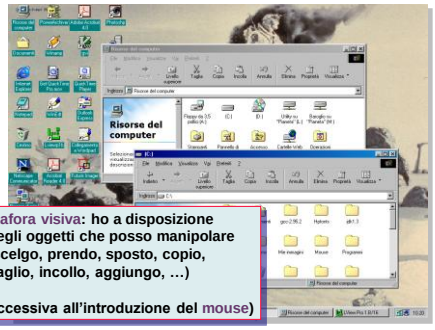

## Interfaccia testuale




**Metafora linguistica: parlo al computer dicendogli cosa deve fare ed esso mi parla scrivendomi il risultato della elaborazione**

**(precedente l'introduzione del mouse)**

## Interfaccia grafica(GUI)




**Metafora visiva: ho a disposizione degli oggetti che posso manipolare (scelgo, prendo, sposto, copio, taglio, incollo, aggiungo, …)**

**(successiva all'introduzione del mouse)**

## Sistema operativo: Kernel

- Kernel (nocciolo, nucleo): programmi per le funzioni base del calcolatore
- Da 100 Kilobyte a 100 Megabyte
- Kernel suddiviso in moduli
- Ogni modulo ha una funzione diversa
- Funzioni piu' importanti:
  - gestione processori
  - gestione processi
  - gestione memoria (principale e secondaria)
  - gestione dispositivi di I/O

## Gestione memoria secondaria: il File Manager

- Gestisce la memoria secondaria
  - Associa un nome di file ad una parte dello spazio di memoria(si dice che tale spazio è "allocato" ovvero riservato al file)
  - Fornisce metodi per accedere ai file e per proteggere gli stessi da accessi o modifiche non autorizzate
  - Rende trasparente(ovvero cela agli occhi dell'utente) la struttura fisica della memoria e la maniera in cui sono memorizzati i file
  - Ottimizza l'occupazione di memoria
- Organizzazione dei file in cartelle (directory), gerarchia di cartelle

## File

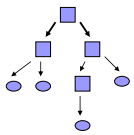
- Unica unita' logica di informazione usata dal SO
- Fisicamente:
  - sequenza di byte che contiene informazioni omogenee
  - Es.: programma, testo, dati simili, …
- Tutti i dati vengono suddivisi in file
- I file vengono memorizzati nelle memorie di massa

## Organizzazione dei file in memoria secondaria

- Organizzazione gerarchica
- Non vi è nessuna relazione tra questa organizzazione logica e la posizione fisica delle informazioni sulle memorie di massa
- Directory: gruppo di file e altre directory

## Gerarchia ad albero

- Albero rovesciato
- Nodi e collegamenti padre-figlio tra nodi
- Nodo: file o directory
- Nodi divisi per livelli
- Collegamenti tra nodi di livelli vicini: nodo sopra = padre, nodo sotto = figlio
- Ogni nodo ha un solo padre
- Padre piu' in alto = radice
- I nodi file non hanno figli
- Per individuare un file:
  - Cammino (path) assoluto o relativo
  - lista di nomi di cartelle

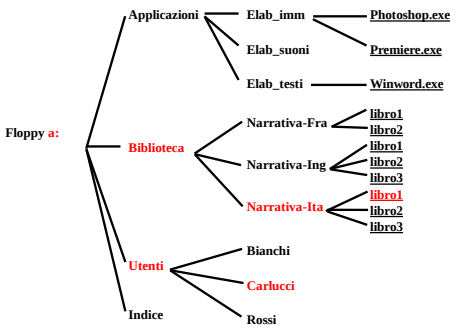



## Operazioni su file

- Creazione
- Apertura
- Chiusura
- Cancellazione
- Copia
- Rinomina
- Visualizzazione
- Lettura
- Scrittura
- Modifica
- …

## Individuare un file nel File System

- Se non esistesse la strutturazione in directories, tutti i file potrebbero essere identificati mediante il loro nome
- Nel caso di un'organizzazione gerarchica a più livelli il nome non è più sufficiente ad identificare il file (possono esistere diversi file con lo stesso nome situati in directories differenti)
- Per individuare un file o una directory in modo univoco si deve allora specificare l'intera sequenza di directories che lo contengono, a partire dalla radice dell'albero




# Path

- Ad esempio il file *libro1* di narrativa italiana è univocamente identificato dalla sequenza:

  *A:\Biblioteca\Narrativa-Ita\libro1*

- La directory *Carlucci* di *Utenti* è identificata dalla sequenza:

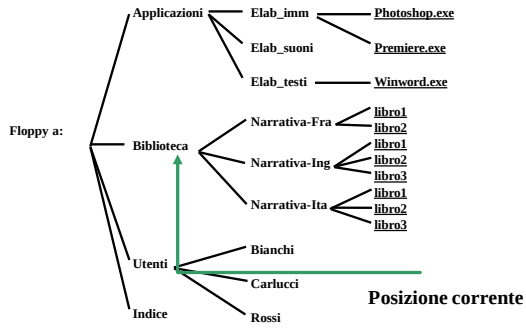
  *A:\Utenti\Carlucci*

  il carattere "\"viene usato come separatore.

- Una sequenza di questo tipo può essere vista come il *cammino* che si deve compiere per raggiungere un determinato file a partire dalla radice dell'albero, ed è chiamata **path**

# Path assoluto,corrente,relativo

- Si dice "path assoluto" il path che specifica la posizione corrente nel File System rispetto al nodo radice
- Si dice "path corrente" il path della cartella (directory) in cui ci si trova o del file aperto(in DOS è quello che compare prima del prompt ">", in Windows lo vedete nella riga in alto della finestra(*window*)).Indica la posizione attuale nel File System
- Si parla di "path relativo" quando si specifica la posizione del file rispetto alla posizione attuale nel file system ovvero rispetto al path corrente



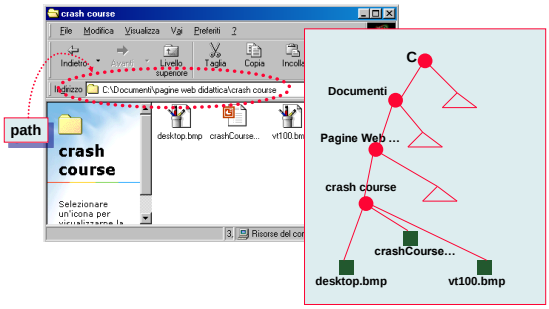
- Se la posizione corrente è *A:\Biblioteca,* il file *libro1* di narrativa italiana è univocamente identificato dalla sequenza:

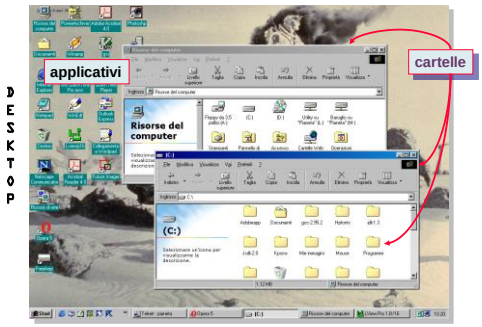
  *Narrativa-Ita\libro1*

- Se la posizione corrente è *A:\Utenti,* la directory *Carlucci* è identificata dal **path relativo**:

  *Carlucci*

## Path/directory




## Assegnare un nome ad un file

- Il nome è generalmente composto da due parti
  - il nome vero e proprio (si possono usare caratteri alfanumerici in numero dipendente dal SO)
  - l'estensione (una sequenza di caratteri che aiuta ad identificare il tipo di contenuto del file)
  - nome ed estensione sono separati da un ***punto***
  - Il nome è obbligatorio mentre l'estensione è opzionale (ma fortemente consigliata)

## Assegnare un nome ad un file

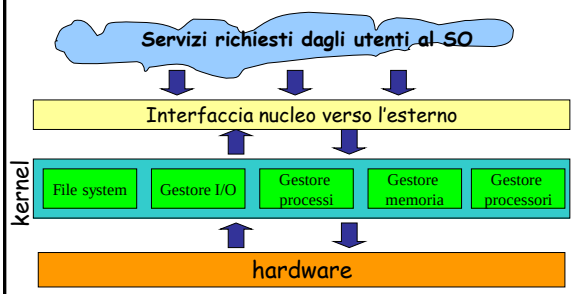
Pippo.txt

- Pippo è il nome
- txt è l'estensione che fornisce informazioni circa il tipo di file(che tipo di dati contiene e in che maniera sono memorizzati ovvero in che formato) ed eventualmente sull'applicazione con cui è stato creato o che serve per leggerlo
- Es
  - .txt ->file di testo
  - .doc -> file Word (testo,immagini,etc…)
  - .pdf -> il file può essere letto con Acrobat Reader
  - .mp3 ->file musicale

## Modo Supervisore e modo Utente

- La memoria è suddivisa in:
  - Memoria di modo S (supervisore): contiene i programmi di sistema operativo e vengono create le strutture dati da esso utilizzate
  - Memoria di modo U (utente): contiene i programmi utente
- Se il processore sta eseguendo un processo utente:
  - Lo si dice attivo in modo utente (in modo U)
  - Può accedere soltanto alla memoria di modo U
- Se il processore sta eseguendo un processo di sistema:
  - E' attivo il kernel del SO
  - Si dice attivo in modo supervisore (in modo S)
  - Può accedere alla memoria di modo S e di modo U
  - Può eseguire istruzioni "privilegiate"

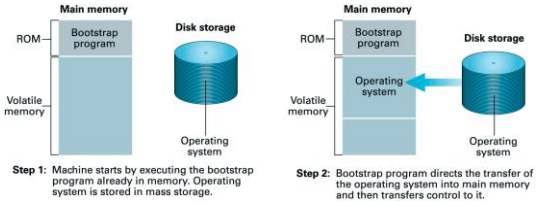
## Struttura del SO




## Bootstrap: avvio del SO

- All'inizio, la CPU ha in indirizzo fissato nel registro contatore di programma(Program Counter)
- Indirizzo della prima istruzione del programma di bootstrap nella ROM (Read Only Memory)
- Le istruzioni contenute nel programma di bootstrap trasferiscono una parte prestabilita della M di massa in M principale (kernel del SO)
- "Salta" alla prima istruzione del kernel del SO

## Bootstrap




## Gestione processi e processori

- Scheduler (*gestore processi*): decide quali processi mandare in esecuzione
- Dispatcher (*gestore processori*): decide per quanto tempo ogni processo ha a disposizione una data CPU(se ne esistono più di una) prima che sia "messo in attesa" forzatamente ed un'altro processo sia mandato in esecuzione al suo posto (per realizzare il multi-tasking)

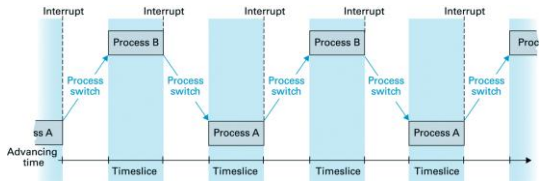
## Gestione dei processi

- Scheduler: tiene traccia di tutti i processi
- Tabella dei processi in M principale
- Nuova richiesta di esecuzione di un programma
  ➔ nuovo elemento nella tabella dei processi
  - Area di M assegnata per quel processo
  - Priorita'
  - Pronto (per essere eseguito) o in attesa (di qualche evento)

## Dispatcher

- Divide il tempo in quanti (< 50 millisec.)
- Da' un quanto ad ogni processo, uno alla volta
- Alla fine del quanto prima di passare al prossimo processo, la CPU esegue il programma di gestione delle interruzioni
  - Aggiorna la tabella dei processi
  - Salva lo stato (registri, celle di M, ...)
  - Sceglie un altro processo dalla tabella che si trovi nello stato "pronto"

## Time-sharing (condivisione di tempo)



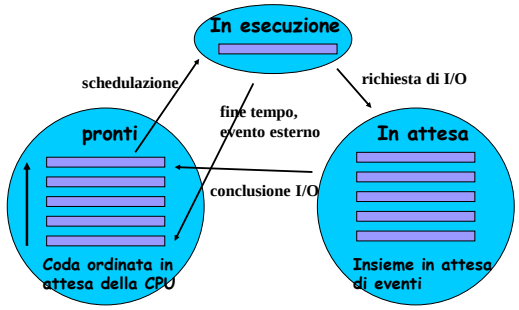

## Attesa

- Se il processo ha bisogno che siano effettuate delle operazioni(es. Operazioni di I/O) per poter procedere,l'esecuzione del medesimo viene interotta in attesa che siano disponibili le risorse richieste.
- La CPU infatti rimanerebbe inutilizzata durante questo periodo per cui lo scheduler mette in stato di attesa il processo corrente e il dispatcher sceglie dalla tabella un nuovo processo tra quelli "pronti" per l'esecuzione
- Quando l'operazione sara' finita, lo scheduler dichiarera' di nuovo "pronto" per l'esecuzione il processo

## Riassumendo ...

- I processi sono messi in una coda dei "pronti" per l'esecuzione
- Il processo in cima alla coda ha a disposizione la CPU per un quanto di tempo fissato stabilito in base al tipo di processo(CPU Bound o Memory Bound) e alla sua importanza
- Quando il tempo finisce, viene interrotto e messo in fondo alla coda
- Se richiede un'operazione di I/O, va nel gruppo dei processi in attesa, quando l'operazione finisce viene rimesso in coda per la CPU
- La CPU va al primo in coda (o altre strategie)
- Durante la vita di un processo, vari stati:
  - In esecuzione: sulla CPU (solo un processo alla volta)
  - Pronto: pronto per l'esecuzione
  - In attesa: in attesa di un evento (es.: fine di op. di I/O)

## Transizioni tra stati di un processo



## Gestione dell'I/O

- Gestione delle periferiche
- Rende trasparenti le caratteristiche fisiche delle periferiche attraverso i drivers
- Coordina l'accesso alle risorse condivise

## Driver (di periferica)

- Il driver, qui inteso, è un software che fornisce una interfaccia tra SO e hardware della periferica
- In pratica, se devo stampare un documento, il driver si preoccupa di tradurre il comando "stampa la pagina corrente" nella serie di input binari opportunamente temporizzati da fornire al controller della stampante
- Il controller della stampante interpreta questi input generando una sequenza di segnali elettrici inviati ai vari componenti meccanici ed elettronici in maniera da compiere l'azione desiderata
- Se durante l'operazione si verificano problemi, il controller della stampante asserisce dei bit che segnalano l'errore e il driver lo segnala a sua volta al SO
- Il driver è realizzato dal costruttore della periferica e fornito insieme ad essa nella forma di un CD di installazione, eventualmente insieme ad un programma applicativo che permette all'utente di sfruttare le caratteristiche più avanzate della periferica.

## Gestione della memoria principale

- Se si lancia un solo programma alla volta(sistemi mono-tasking) il SO non ha molto da fare:semplicemente carica il programma in memoria,lo manda in esecuzione ed una volta terminato lancia il prossimo
- Se però si vuole che più programmi vengano eseguiti contemporaneamente(dalla macchina virtuale) allora la situazione si complica notevolmente...
- Tutti i programmi infatti richiedono memoria, ma quest'ultima è limitata, bisogna cercare di accontentare tutti riservando ai programmi in esecuzione memoria in base alle effettive esigenze...
- ...è necessario "allocare" memoria e "rilasciare" lo spazio di memoria non utilizzato affinchè possa essere utilizzato da un'altro programma

## Rilocazione e paginazione

- La gestione concorrente di molti processi, comporta la presenza di molti programmi in memoria centrale
- Per caricare i programmi in memoria è necessario *rilocarli* (trasformare gli indirizzi "logici", presenti nei programmi, in indirizzi "fisici"(ovvero indirizzi di locazioni della RAM))
- *Paginazione*:
  - La memoria centrale è considerata dal gestore della memoria come **partizionata in pagine**
  - Ciascuna di queste è un'area di memoria contigua, di **dimensione fissa**
  - Dati e programmi vengono partizionati in pagine e allocati in un numero intero di pagine, **non necessariamente contigue**

## Segmentazione

- *Segmentazione*:
  - Durante la compilazione, il programma è frazionato in parti che svolgono differenti funzioni
    - Per es. si possono separare i dati dalle istruzioni
  - E' una *partizione logica* del programma.
  - Consente al gestore della memoria di caricare i segmenti che compongono il programma stesso in maniera indipendente
  - Mentre le pagine hanno lunghezza fissa, i segmenti, hanno lunghezza variabile
- La segmentazione e la paginazione non sono tecniche alternative. Spesso sono applicate contemporaneamente(nel senso che i segmenti sono divisi in pagine)

## La memoria virtuale

- In entrambi i casi, il gestore della memoria offre al programma applicativo la visione di una *memoria virtuale*
- La memoria virtuale è maggiore di quella fisica:
  - E' possibile allocare più pagine o segmenti di quelle che possono stare nella memoria fisica allo stesso tempo
  - Le pagine o i segmenti che non sono al momento caricate nella memoria fisica rimango disponibili nella *memoria di massa*
- La gestione della memoria è coordinata con la gestione dei processi:
  - Quando un processo è eseguito, le pagine o i segmenti che sono al momento in esecuzione o che contengono i dati attualmente indirizzati devono essere caricati in memoria
  - Se una pagina o un segmento necessario al programma non è presente, il processo deve essere sospeso per consentirne il caricamento

## Riassumendo...

- La memoria virtuale serve in quanto lo spazio di memoria richiesto e' maggiore di quello fisico
- Programmi e dati vengono spostati tra memoria principale e memoria di massa per avere in ogni momento quello che serve
- Lo spazio richiesto e' diviso in pagine (qualche Kbyte)
- Solo le pagine richieste per l'esecuzione del programma sono caricate nella RAM, le altre rimangono nella memoria di massa
- Vengono caricate dalla memoria di massa a seconda della necessita' e salvate nuovamente sulla memoria di massa se lo spazio sulla RAM è insufficiente all'esecuzione di un programma
  - Il SO si preoccupa di eseguire tutte le operazioni necessarie(rilocazione degli indirizzi,caricamento in memoria,...) necessarie a svincolare il codice di un programma dalla sua collocazione in memoria
  - Funziona come se si avesse piu' memoria di quella reale

## Swapping

- L'assegnazione della memoria è "dinamica"(in contrapposizione a "statica"=indipendente dal tempo e dallo stato del sistema), ovvero dipende dallo stato del sistema in quell'istante
- Dipende ad esempio da:
  - quali processi devono essere eseguiti,
  - il numero di pagine richieste durante l'esecuzione,
  - il grado di priorità di un processo rispetto ad un altro in relazione ad esempio alla presenza di vincoli temporali sulla loro esecuzione(processi real-time o interattivi),
  - lo stato del processo all'istante considerato(in esecuzione,in attesa,pronto per l'esecuzione)
  - etc...
- L'area di memoria di massa che realizza la memoria virtuale si chiama area di swap
- Questo perchè le pagine vengono scambiate tra quest'area e la RAM(in inglese scambiare=*swapping*)