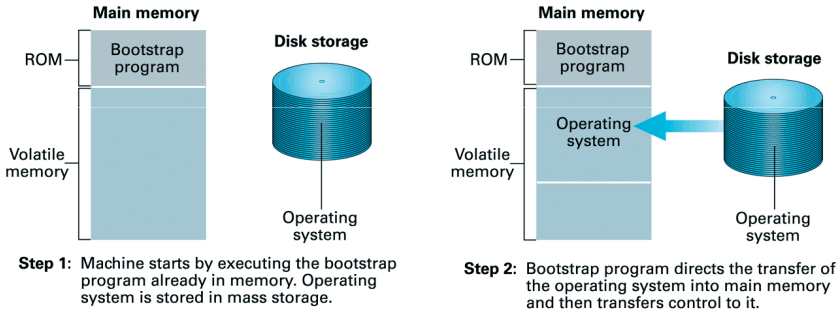
Bootstrap

- **All'accensione** di un calcolatore vengono attivati programmi di diagnostica scritti nella ROM (Read Only Memory) che verificano l'assenza di guasti
- Poi viene attivato il programma di **bootstrap** che e' sempre memorizzato nella ROM
- Il bootstrap trasferisce una parte prestabilita della M di massa (disco rigido, CD o floppy) in M principale (kernel del SO)
- Poi l'utente puo' impartire comandi al SO attraverso l'interfaccia utente (tastiera, mouse…)

Bootstrap



Shutdown

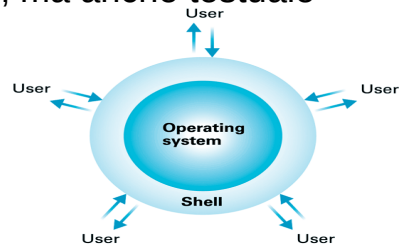
- Quando il calcolatore viene **spento** una fase di **shutdown** si assicura che tutte le informazioni che temporaneamente il SO ha caricato su memoria volatile vengano memorizzate su memoria di massa prima che l'energia venga tolta dalla macchina

Struttura di un SO

- Elementi principali di un SO:
  - Shell (interfaccia SO e utente)
  - Kernel (insieme di programmi che realizzano le funzioni di base di un calcolatore)

Sistema operativo: shell

- Shell (guscio): **interfaccia tra SO e utenti**
- E' il programma che permette agli utenti di comunicare con il sistema e di avviare I programmi
- Di solito grafica (GUI – Graphical User Interface), ma anche testuale



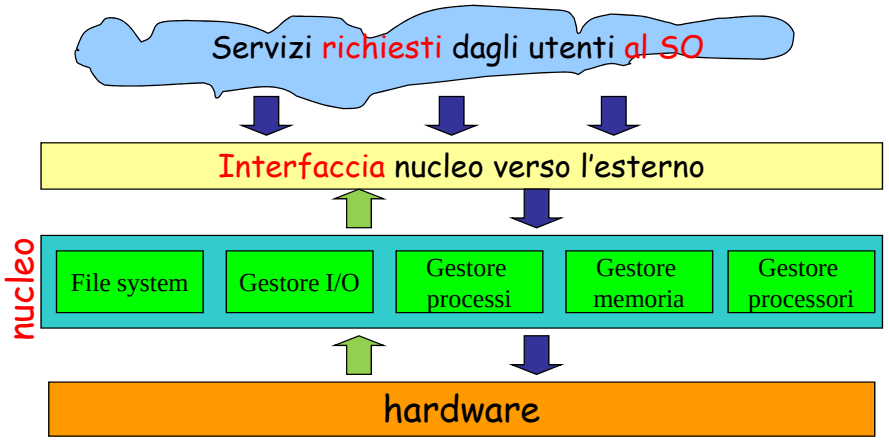
Sistema operativo: kernel (1)

- Kernel (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 processi
  - gestione processori
  - gestione memoria (principale e secondaria)
  - gestione dispositivi di I/O

Sistema operativo: kernel (2)

- Funzioni piu' importanti:
  - **gestione processi:** controlla la sincronizzazione, l'interruzione e la riattivazione dei programmi in esecuzione
  - **gestione processori**: assegna il processore ai programmi che devono essere eseguiti (e gestisce la cooperazione tra varie CPU nel caso di piu' calcolatori)
  - **gestione memoria** (principale e secondaria): gestisce la collocazione dei dati e dei programmi nella RAM, la memorizzazione e il reperimento delle informazioni sulla memoria secondaria
  - **gestione dispositivi di I/O** (gestisce l'accesso e il controllo dei dispositivi periferici)

Struttura del SO



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.: due utenti che usano Word per scrivere due documenti diversi

Esempio

Fare un dolce con una certa ricetta

- 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
- Il processo interrompe l'esecuzione, prende nota del punto (riga della ricetta, ...) e gestisce l'evento (va al telefono e risponde)
- Alla fine della telefonata, l'esecuzione riprende e viene conclusa
- Dolce = output

Processi e SO

- Un SO puo' gestire simultaneamente piu' programmi
- Ogni programma puo' essere fermato e fatto ripartire piu' volte
- Quindi ogni SO deve avere una struttura (detta **processo**) in grado di mostrare, istante per istante, lo stato di avanzamento di ciascun programma

Obiettivi della gestione dei **processi**

- Esecuzione e avvicendamento dei processi
- Sincronizzazione e comunicazione tra diversi processi
- Al fine di ottimizzare
  - Condivisione delle **risorse fisiche** (CPU, memoria)
  - Condivisione **risorse logiche** (dati)
  - **Modularita'** (piu' processi che comunicando tra loro svolgono l'intero compito)
  - **Incremento della velocita'** (suddivisione di un lavoro in sottolavori eseguiti parallelamente da piu' processori)

Gestione dei processi: Scheduler (1)

- **Tiene traccia di tutti i processi**
- A ogni processo e' associata una Tabella (detta tabella di processo) in M Principale
  - Identificativo
  - Indirizzo prossima istruzione
  - Contenuto registri CPU
  - Stato corrente (Esecuzione, Bloccato, Pronto)
  - Informazioni per gestire la M (collocazione in M centrale di dati e programmi)
  - Priorita'
  - I/O (richieste inoltrate, dispositivi allocati al processo)
  - Accounting (ammontare di CPU time gia' speso)

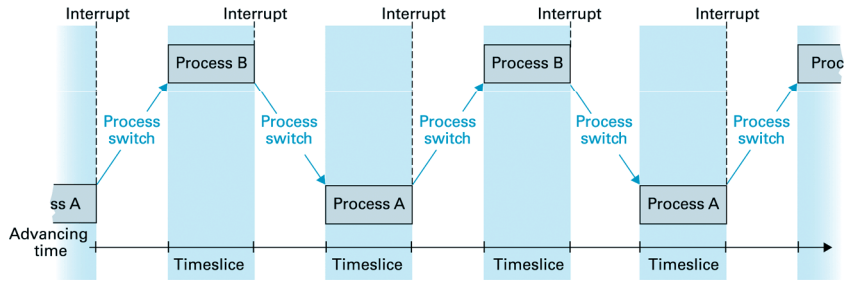
Gestione dei processi: Scheduler (2)

- Usando questa tabella si possono compiere le seguenti operazioni
  - Creazione (un processo puo' crearne altri)
  - Cambio di priorita'
  - Blocco
  - Assegnamento della CPU
  - Terminazione

Gestione dei **processori**: Dispatcher

- Divide il tempo in quanti (< 50 millisec.)
- Da' un quanto ad ogni processo, uno alla volta
- Alla fine del quanto, segnale che passa la CPU ad un altro processo pronto
- 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

Time-sharing (condivisione di tempo)



Gestione della coda

- Un processo in esecuzione puo' essere **interrotto** per vari motivi:
  - Quanto esaurito
  - Esecuzione completata
  - Subentra un processo con maggiore priorita'
  - Ha fatto una richiesta di un'operazione di I/O → entra nella lista dei processi bloccati
- → modifica del program counter al vettore di interrupt (diverso per ogni evento)
- Politica di scelta del successivo processo da eseguire:
  - **FIFO**: First in First out
  - **SJF**: Shortest Job First
  - **Deadline** (si esegue prima il processo con la scadenza piu' vicina)
  - **SRT**: Shortest Remaining Time

Attesa

- Se il **processo** richiede operazioni **ad altri dispositivi** (es. Operazioni di I/O), la CPU rimarrebbe inutilizzata ➔ lo scheduler mette il **processo in stato di attesa**, e il dispatcher sceglie un **nuovo processo** tra i pronti dalla tabella
- Quando l'operazione sara' finita, lo scheduler dichiarera' di nuovo **pronto** il processo

Riassumendo ...

- I processi sono messi in una **coda dei pronti**
- Il processo in cima alla coda ha a disposizione la CPU per un **quanto di tempo**, fissato e uguale per tutti
- 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**: quando usa la CPU (solo un processo alla volta)
  - **Pronto:** quando aspetta la CPU
  - **Bloccato:** in attesa di un evento (es.: fine di op. di I/O)