Processi

Definizione (Processo)

Un **processo** *è un'attività controllata da un* **programma** *che si svolge su un* **processore**.

- È il modo in cui un programma viene eseguito **nel tempo**.
  - Programma: entità statica.
  - Processo: entità dinamica.
- Ciascun processo ha uno **stato**, composto da un certo numero di informazioni
  - Immagine di memoria
  - Immagine nel processore
  - Stato di avanzamento

Stato di un processo

Definizione (Immagine di memoria)

*L'***immagine di memoria** *di un processo è l'insieme di informazioni relative al processo mantenute nella memoria principale del calcolatore.*

- **Codice** del programma in esecuzione
- **Dati** su cui sta lavorando
- **Strutture dati del SO** per la gestione del processo

Stato di un processo

Definizione (Immagine nel processore)

*L'**immagine nel processore** di un processo è il contenuto dei registri del processore, che vengono utilizzati come memoria temporanea durante l'esecuzione delle istruzioni.*

- Esempi:
  - **Program counter**: la prossima istruzione da eseguire
  - **Flag**: esito di una operazione della ALU
  - **A**, **B**, ...: operandi

Stato di un processo

Definizione (Stato di avanzamento)

Lo **stato di avanzamento** *descrive l'attuale attività del processo.*

- Esempi di possibili stati di avanzamento
  - **Waiting**: in attesa di qualche evento
  - **Running**: in esecuzione
  - **Ready**: attende di essere eseguito

Multi-programmazione e time-sharing

Definizione (Multi-programmazione)

Tecnica di gestione della CPU secondo la quale durante i **periodi di I/O** *di un processo vengono eseguiti altri processi.*

Definizione (Time-sharing)

Tecnica di gestione della CPU secondo la quale l'esecuzione del processore viene suddivisa in un certo numero di **quanti temporali**; *allo scadere di un quanto, il processo corrente viene interrotto e l'esecuzione passa ad un altro processo.*

- Numerosi vantaggi:
  - Processore non inattivo durante operazioni di I/O (lunghe).
  - Memoria utilizzata al meglio, caricando il maggior numero di programmi possibile.
  - Impressione di esecuzione contemporanea di processi diversi.

Meccanismo degli interrupt

- Implementazione:
  - Tramite un componente del SO detto **scheduler**
  - Invocato ogni volta che una operazione di I/O viene:
    - **iniziata** (da un processo)
    - **terminata** (da un dispositivo)
  - Usa meccanismo degli **interrupt**

→ **richiesta di I/O da parte di un processo**

- interrupt software

→ **completamento di tale operazione**

- interrupt hardware generato dal dispositivo associato

- suddivisione **in quanti temporali**
  - tramite un processo **timer**, che è in grado di generare un interrupt periodicamente
- Per passare da un processo a un altro: **context switch**.

Lo stato di avanzamento dei processi

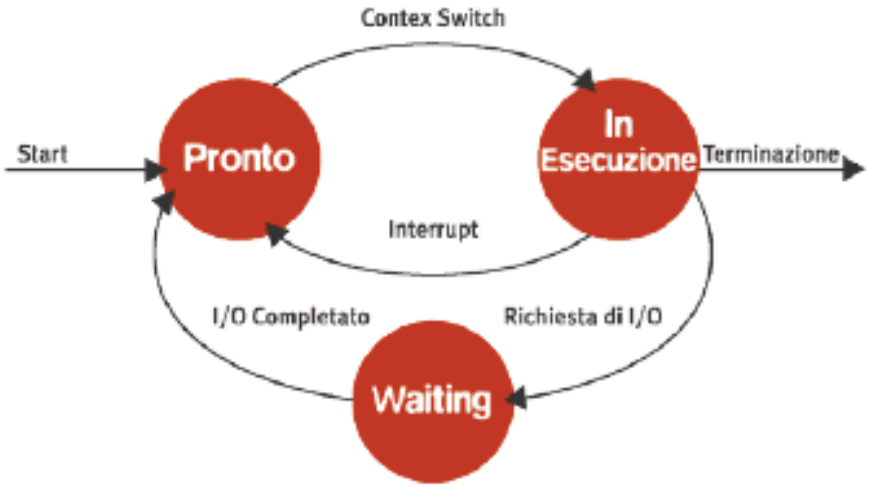
Lo stato di avanzamento dei processi

- **In esecuzione (running)**
  - il processo è in esecuzione
    - 1 CPU ⇒ 1 solo processo in esecuzione per volta
    - multi-core: 1 processo per CPU
  - controllo passa al SO se:
    - interrupt hardware ⇒ *ready*
    - interrupt software ⇒ *waiting*

Lo stato di avanzamento dei processi

- **In esecuzione (running)**
- **In attesa (waiting)**
  - il processo è in attesa di evento esterno (tipicamente, I/O)
    - non può essere eseguito
  - quando l'evento si verifica $\Rightarrow$ *ready*

Lo stato di avanzamento dei processi

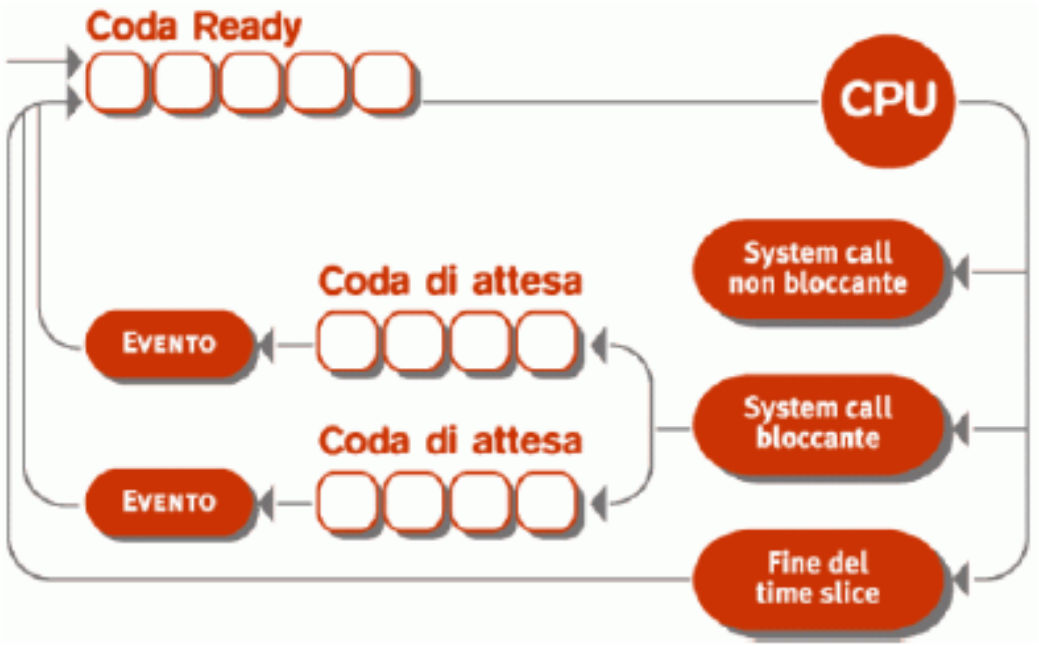


- **In esecuzione (running)**
- **In attesa (waiting)**
- **Pronto (ready)**
  - il processo può essere eseguito, ma
  - la CPU è impegnata da un altro processo
  - il processo attende che il SO lo faccia ripartire ($\Rightarrow$ *running*)

Lo scheduler

Definizione (Scheduler)

Lo **scheduler** *è il componente del sistema operativo che decide di volta in volta quale processo deve essere eseguito.*



Informazioni sui processi attivi (UNIX)

- Il SO tiene traccia di alcune informazioni sui processi, tra cui:
  - Direttorio corrente (**working directory**)
  - Info sui file usati dal processo (**file descriptor table**)
  - ID del processo (**Process ID, PID**)
  - ID del gruppo (**Group ID, GID**)
  - ID del processo "padre" (**Parent Process ID, PPID**)
  - Variabili di ambiente (es: `PATH`)

Come interagire con i processi

- Funzioni del SO per operare con i processi:
  - **Creazione di un processo**: `fork()`
  - **Esecuzione di un programma**: `exec()`
  - **Terminazione**: `exit()`
  - **Sincronizzazione tra processi**: `signal()` e `kill()`
  - **Comunicazione tra processi**: `send()` e `receive()`
- Operazioni comuni per un utente:
  - esecuzione di un programma (es. *double click*)
  - terminazione di un processo (es. *task manager*)

Gestione della Memoria

Gestore della memoria

Definizione (Gestore della memoria)

Il **gestore della memoria** *è il componente del sistema operativo che si occupa di gestire la memoria per conto dei processi.*

- La memoria serve a ogni processo in esecuzione, per contenere **codice** e **dati**.
- La memoria (principale) è una **risorsa limitata** (e costosa):
  - Gerarchia delle memorie
    - Memoria principale (100 €): $\sim$ GB
    - Memoria secondaria (100 €): $\sim$ TB
- Uso di una porzione dell'Hard Disk (**partizione di swap**) per emulare la memoria principale (**memoria virtuale**)

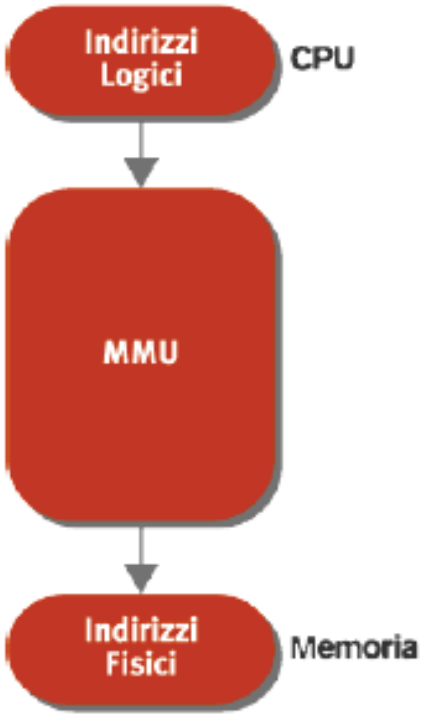
Gestore della memoria

Definizione (Gestore della memoria)

Il **gestore della memoria** *è il componente del sistema operativo che si occupa di gestire la memoria per conto dei processi.*

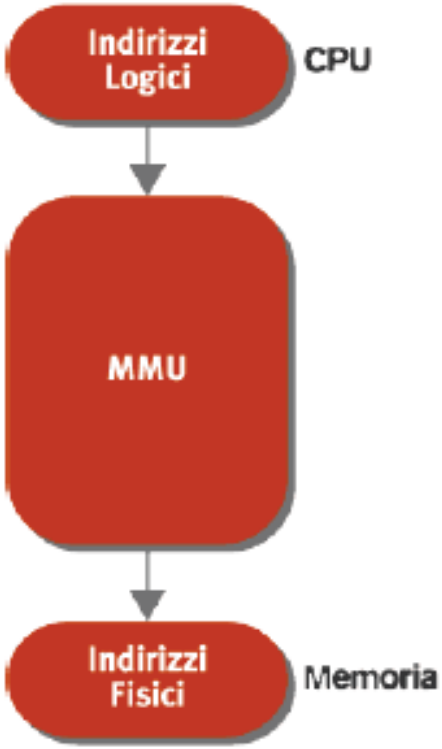
- Compiti del gestore della memoria:
  1. Tenere traccia di quali processi occupano quali porzioni di memoria
     - Uso di **tabelle di allocazione**
  2. Assegnare memoria ai processi che ne facciano richiesta
     - **Allocazione** di **pagine** di memoria
  3. Rilasciare la memoria quando non è più richiesta
     - **Deallocazione** di pagine di memoria

Memoria fisica e memoria logica



- **Spazio di indirizzamento logico**
  - Ogni processo è associato ad uno spazio di indirizzamento logico
  - Gli **indirizzi usati in un processo** sono indirizzi logici
- **Spazio di indirizzamento fisico**
  - Ad ogni indirizzo logico corrisponde un indirizzo fisico della memoria
  - Serve una **funzione di traduzione** da indirizzi logici a indirizzi fisici
  - La Memory Management Unit (MMU) è un dispositivo che esegue tale funzione

Memoria fisica e memoria logica



- Vantaggi di questa traduzione:
  - È meglio **nascondere** ai processi la reale organizzazione della memoria
    - Per **semplificare** il codice
    - → ad es: il codice usa sempre gli indirizzi da 0 a 1000
    - → indipendentemente dal gestore
    - → anche se in RAM non esistono tutti gli indirizzi da 0 a 1000
    - Per **proteggere** la memoria
    - → evitare che un processo interagisca su zone su cui non ha i diritti
  - La MMU può **ottimizzare** l'utilizzo della memoria
    - Il singolo processo non può sapere

File System

Il file system: un'astrazione

- I computer possono utilizzare **diversi media** per registrare in modo permanente le informazioni
  - esempi: dischi rigidi, floppy, nastri, dischi ottici
  - ognuno di questi media ha **caratteristiche fisiche diverse**
- Un **file system** nasconde la complessità dei diversi media proponendo una astrazione:
  - indipendente dal supporto di memorizzazione
  - efficiente
  - conveniente da usare
- **Per l'utente**, un file system è composto da **due elementi**:
  - **file**: l'**unità logica di memorizzazione**;
  - **directory** (**folder**): un insieme di informazioni per **organizzare i file** che compongono un file system

File

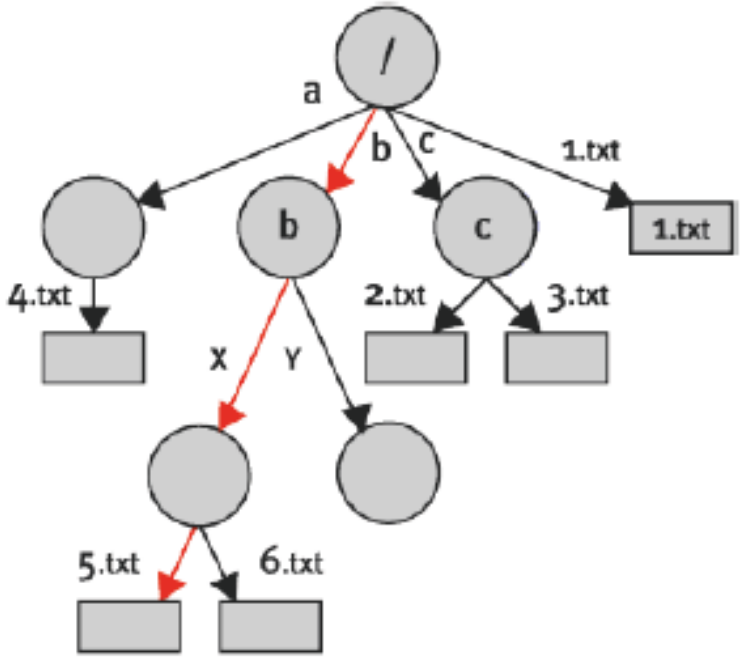
- Ogni file è caratterizzato da un insieme di **attributi**:
  - **nome** del file
  - informazioni su **locazione** e **dimensione**
    - difficile memorizzare su blocchi contigui $\Rightarrow$ frammentazione
    - utilizzo di **indici** per tener traccia dei blocchi occupati
  - informazioni **temporali**: creazione, ultima modifica, ultimo accesso
  - **proprietà**: per associare un file a un proprietario (**owner**)
    - quali sono i **privilegi** dell'owner sul file
    - chi ha il permesso di **leggere**, **modificare**, **eseguire** un file
  - **tipo**: file, immagini, ... Per indicare il tipo:
    - uso di **estensioni** del nome, es. `Lucidi_07.pdf`
    - informazioni **nel contenuto**, es. primi caratteri: `%PDF...`

Come operare sui file

- Funzioni del SO per operare con i file:
  - **Creazione di un file**: `create()`
  - **Apertura/chiusura di un file**: `open()`/`close()`
  - **Lettura/modifica**: `read()`/`write()`
  - **Posizionamento**: `lseek()`
  - **Cancellazione**: `unlink()`
  - **Modifica degli attributi**: `chmod()`, `chown()`
- L'**apertura** serve per caricare tutte le informazioni sulla gestione di un file in alcune strutture dati del SO
- La **lettura** e **scrittura** avvengono tramite l'astrazione di una **testina di lettura**
- Operazioni comuni per un utente:
  - esecuzione di un programma (es. *double click*)
  - spostamento di un file (es. *drag & drop*)
  - visualizzazione di informazioni sul file (es.  + )

Directory

- Concetto fondamentale per **organizzazione gerarchica**
  - Struttura **ad albero**, parte da un direttorio **radice**
  - Directory ⇒ **nodi**, file ⇒ **foglie**
  - Percorso (**path**) per raggiungere un file:
    - Relativo: a partire dal direttorio corrente (.)
    - Assoluto: a partire dal direttorio radice (/)



Come operare sulle directory

- Funzioni del SO per operare con i file:
  - **Creazione** di una directory: `mkdir`
  - **Modifica del direttorio corrente**: `cd`
  - **Visualizzazione del direttorio corrente**: `pwd`
  - **Modifica** di una directory $\Leftrightarrow$ **creazione/cancellazione di un file**: `create()`, `unlink()`
  - **Modifica degli attributi di una directory** `chmod()`, `chown()`
  - **"Esecuzione" di una directory $\Leftrightarrow$ transito** `cd`
  - **Linking di un file**: `ln`
- Operazioni comuni per un utente:
  - modifica del direttorio corrente (es. *double click*)
  - rinominazione di un direttorio (es. *click*)
  - visualizzazione di informazioni sulla directory (es. ⌘ + I)