Tecniche Automatiche di Acquisizione Dati

Sistemi operativi

Cosa sono i sistemi operativi

- I sistemi operativi sono dei programmi software che svolgono le funzioni di interfaccia tra l'hardware e l'utilizzatore.
- mettono a disposizione l'ambiente in cui si eseguono i programmi.
- Poiché sono organizzati secondo criteri differenti, la loro struttura interna è molto diversa.
- I SO possono essere descritti da vari punti di vista:
  - dai servizi che mettono a disposizione,
  - dalle interfacce che forniscono,
  - dal funzionamento dei sottosistemi e dalla loro interazione

Processi

- Il processo è un programma in esecuzione e necessita di alcune risorse del sistema: tempo CPU, memoria, file e dispositivi di I/O
- Un processo include:
  - il *program counter*
  - lo *stack*
  - una sezione dati
- Il sistema operativo è responsabile di:
  - Creazione e cancellazione di processi
  - Sospensione e riattivazione
  - Sincronizzazione comunicazione e gestione dei “deadlock”

Contesti

- Ciascun processo è definito nel S.O. da una collezione di dati chiamata blocco di contesto che include informazioni su:
  - Lo stato e la statistica del processo
  - L'uso della memoria principale e di riserva
  - Apertura di file
  - Apertura di dispositivi
  - Privilegi
  - Statistiche di utenza (accounting)

Lo stato dei processi

- Mentre viene eseguito un processo cambia **stato:**
  - **New** (nuovo): Il processo viene creato.
  - **Running** (in esecuzione): Le istruzioni vengono eseguite.
  - **Waiting** (in attesa): Il processo è in attesa di un evento.
  - **Ready** (pronto): Il processo è in attesa di essere assegnato ad un processore.
  - **Terminated** (terminato): Il processo ha terminato la propria esecuzione.

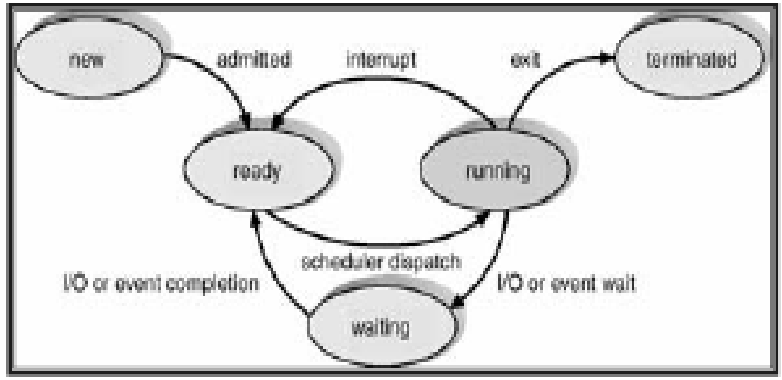
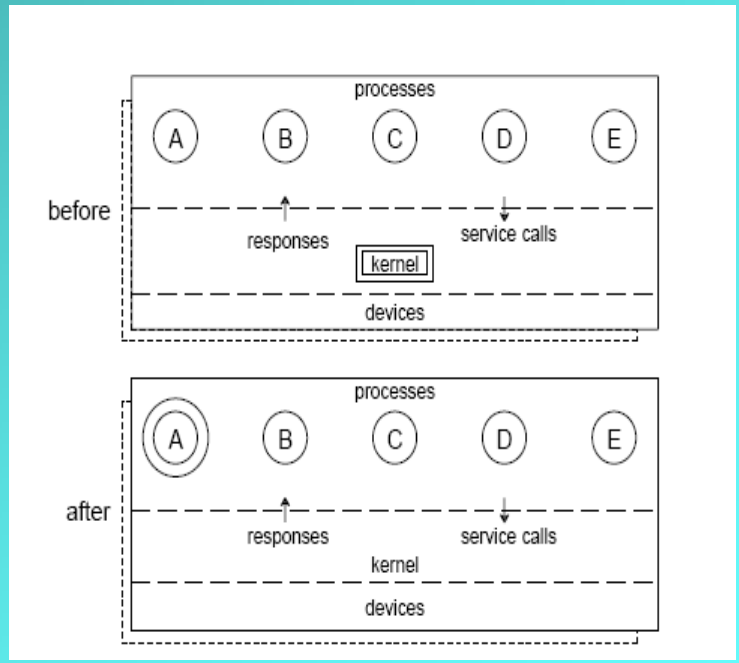


Diagramma degli stati di un processo

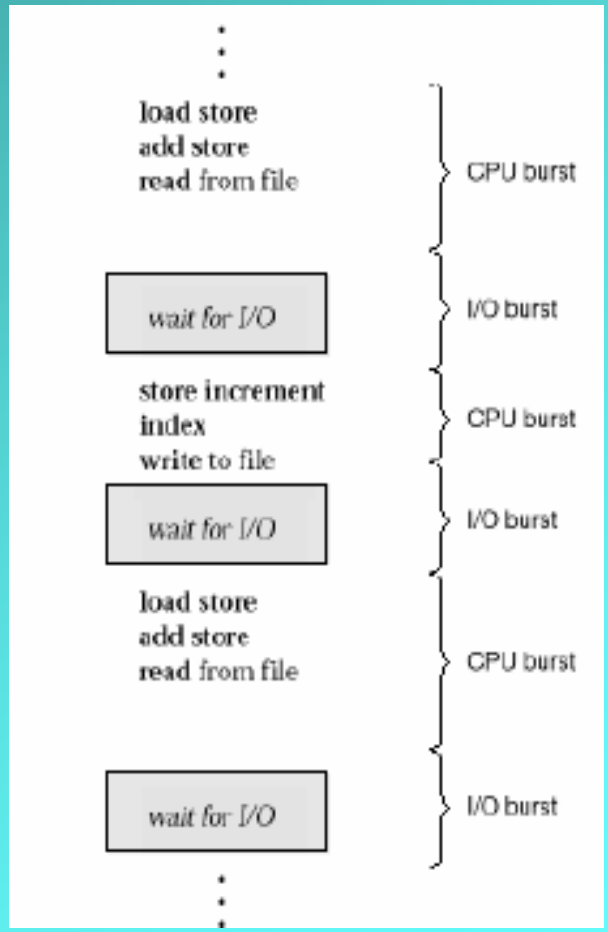
Il kernel

- Il nucleo centrale del sistema operativo è il **kernel**, ovvero un programma di controllo che reagisce agli interrupt dai dispositivi di I/O e alle richieste di servizi dai processi.
- Il kernel è un residente permanente del sistema operativo ed è un processo esso stesso.
- Il passaggio tra l'esecuzione del kernel all'esecuzione di un altro processo si chiama *switch di contesto* perché l'hardware deve passare dal contesto in cui "gira" kernel a quello del processo.



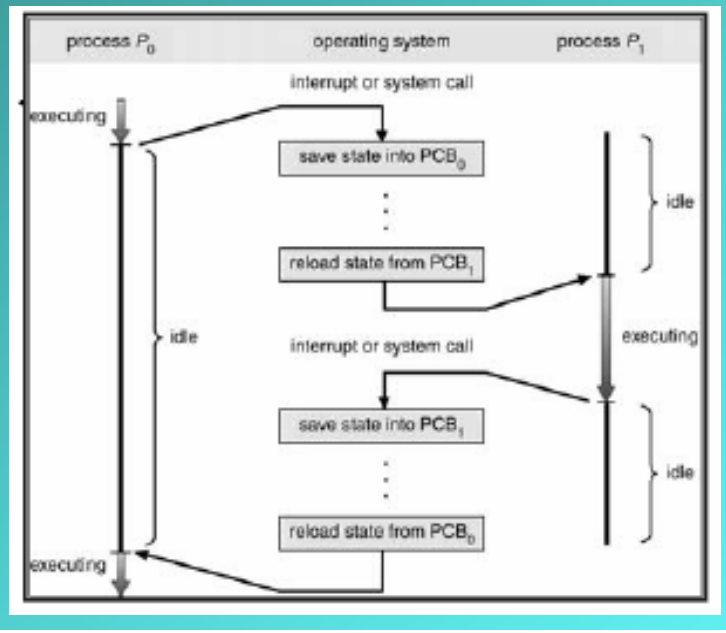
Multiprogrammazione

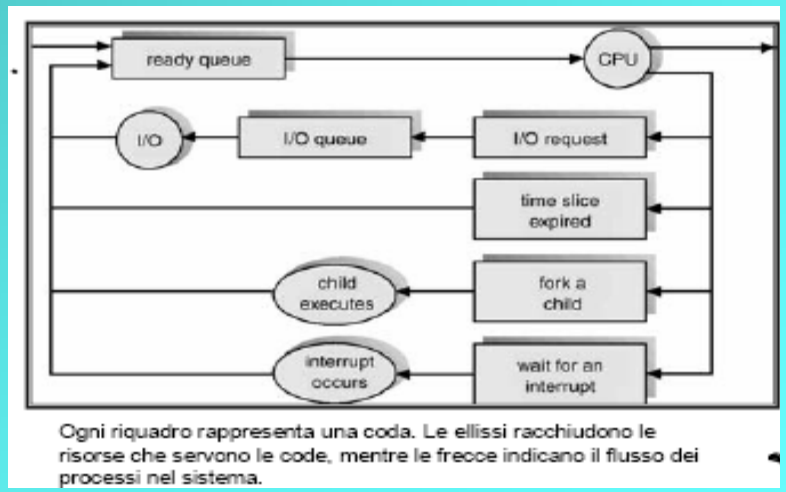
- Il massimo impiego della CPU è ottenuto con la multiprogrammazione.
- *Ciclo di CPU–I/O burst:* L'esecuzione di un processo consiste di *cicli* di esecuzione da parte della CPU ed attese di I/O.
- In generale l'esecuzione inizia con una sequenza di operazioni di elaborazioni (*CPU burst*), seguita da una sequenza di operazioni di I/O (*I/O burst*), in maniera ciclica.



Lo scheduling

- Lo scheduling è la gestione del tempo ed il suo compito è quello di servire i vari processi che competono per le risorse del sistema tramite l'assegnazione di unità di tempo (time slices).
- Gestisce anche la commutazione fra i processi memorizzando il blocco di contesto (program context block – PCB).
- I blocchi di contesto vengono memorizzati in liste differenti a seconda dello stato del processo. Queste liste vengono dette **code** (queues).





Ogni riquadro rappresenta una coda. Le ellissi racchiudono le risorse che servono le code, mentre le frecce indicano il flusso dei processi nel sistema.

Lo scheduler

- Se la CPU passa nello stato d'inattività, il sistema operativo sceglie, fra i processi in memoria pronti ad essere eseguiti, un processo cui allocare la CPU.
- Lo scheduler della CPU deve prendere una decisione quando un processo…
  1. passa da stato **running** a stato **waiting** (richiesta di I/O o attesa terminazione di un processo figlio);
  2. passa da stato **running** a stato **ready** (interrupt);
  3. passa da stato **waiting** a stato **ready** (completamento di un I/O);
  4. termina.
- Se lo scheduling interviene solo nei casi 1 e 4, si dice che lo schema di scheduling è **senza prelazione** (*non–preemptive)*.
- Altrimenti si ha uno schema **con prelazione** (*preemptive).*

Il Dispatcher

- Il modulo allocatore (*dispatcher*) passa il controllo della CPU al processo selezionato dallo scheduler a breve termine; il dispatcher effettua:
  - Cambio di contesto
  - Passaggio a modo utente
  - Salto alla posizione corretta del programma utente per riavviarne l'esecuzione
- *Latenza di dispatch*: è il tempo impiegato dal dispatcher per sospendere un processo e avviare una nuova esecuzione.

Politiche di scheduling

- Esistono diversi algoritmi di scheduling che vengono valutati secondo vari parametri quali:
  - utilizzo CPU,
  - throughput,
  - tempo di completamento,
  - tempo di attesa,
  - tempo di risposta

Esempi di scheduling

- Alcuni esempi di algoritmi di scheduling sono:
  - Scheduling in ordine di arrivo (First Come First Served – FCFS)
  - Scheduling per brevità (SJF): si associa a ciascun processo la lunghezza della successiva sequenza di operazioni della CPU. Si opera lo scheduling in base alla brevità di tale sequenza. È necessaria una stima del successivo tempo di burst.
  - *Shortest–Remaining–Time–First* (SRTF): è un SJF con prelazione. Quando arriva un processo con tempo di burst inferiore al tempo rimanente al processo in corso, il nuovo processo prende la CPU.
  - Scheduling per priorità: A ciascun processo viene assegnato un valore intero di priorità. Viene eseguito il processo a priorità più alta. Dà luogo a problemi di blocco indefinito se non si assegna un criterio di crescita della priorità con il tempo di attesa.
  - Round Robin: algoritmo circolare. A ciascun processo viene assegnato un quanto *q* di tempo (10 – 100ms) dopo il quale viene accodato alla ready queue. Nessun processo attende più di (n-1)*q* unità di tempo.

S.O. e Memoria

- Il sistema operativo deve:
  - Tener traccia delle parti di memoria utilizzate e da chi lo siano
  - Decidere quali processi caricare in memoria quando ci sia lo spazio
  - Allocare e deallocare lo spazio di memoria in base alle necessità.
  - Gestire l'associazione fra le pagine di memoria "virtuale" e le pagine di memoria fisica.

Memoria Virtuale

- Ogni processo ha la sua memoria virtuale che è la memoria “percepita” dal processo
- La memoria virtuale può essere più grande di quella fisica, ed in genere lo è di molto.
- Spesso il sistema operativo divide la memoria virtuale in *segmenti* o in *pagine*, ciascuno con il proprio insieme di indirizzi detto offset.
- I processi hanno a che fare solo con gli indirizzi virtuali, il S.O. deve fornire una traduzione fra indirizzo virtuale e indirizzo fisico (Address Translation)
- Quasi sempre, una regione di memoria virtuale non è *contigua* nello spazio di indirizzamento fisico.

Gestione del filesystem

- Il Sistema operativo è responsabile della organizzazione dei dati in file e cartelle : il filesystem.
- I file sono collezioni di dati correlate definite dal loro creatore ed il sistema operativo deve fornire i servizi per:
  - Creare e cancellare file
  - Manipolare file e cartelle (directories)
  - Associare file a dispositivi di memoria di massa

S.O. e dispositivi di I/O

- Il sistema operativo è responsabile di nascondere all'utente i dettagli hardware dei dispositivi di I/O tramite:
  - Un'interfaccia utente
  - Dei programmi all'interno del kernel (i **driver**) che gestiscono l'hardware.
  - Componenti per la gestione della cache, dell'accodamento e della distribuzione dei dati ai dispositivi.

Protezione

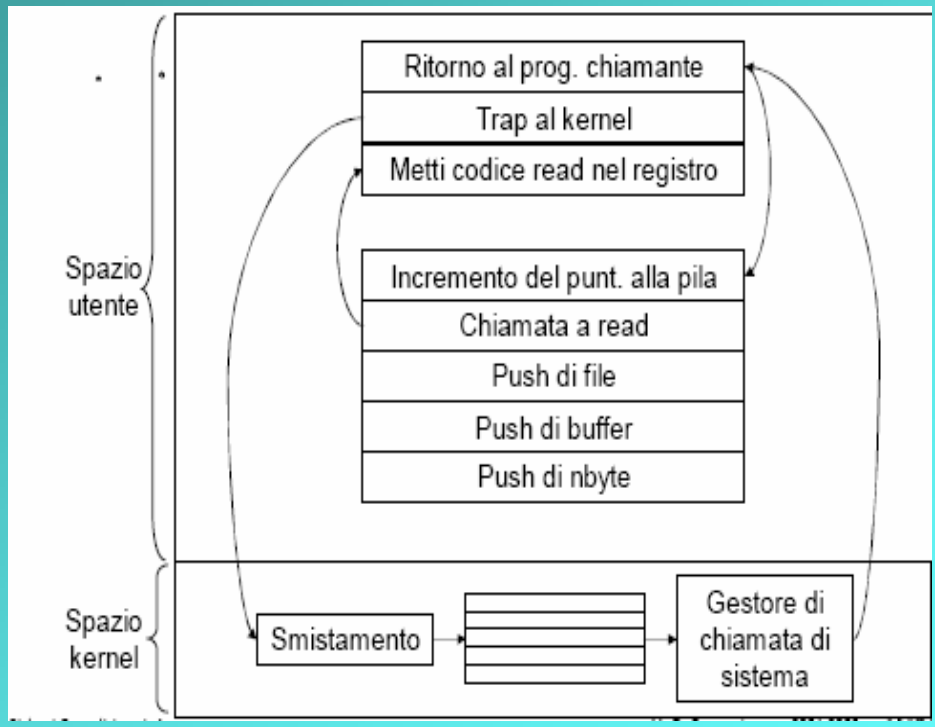
- Il sistema deve gestire l'accesso delle risorse da parte dei vari processi: protezione.
- Ciascun processo può svolgersi solo nel proprio spazio di indirizzamento.
- Il meccanismo di protezione deve:
  - Distinguere tra uso autorizzato e non autorizzato
  - Specificare dei controlli da imporre
  - Fornire una modalità di imposizione

Le chiamate di sistema

- Le chiamate al sistema forniscono l'interfaccia fra unprogramma in esecuzione e il sistema operativo.
- Sono generalmente disponibili come istruzioni in linguaggio assembler.
- Alcuni linguaggi permettono di sostituire il linguaggio assembler per la programmazione dei SO mettendo a disposizione API (Application Programming Interfaces) per le chiamate di sistema direttamente (ad es., C, C++).
- Tre metodi sono impiegati per passare i parametri tra un programma in esecuzione e il sistema operativo.
  - Impiego dei *registri* (passaggio di parametri tramite registri).
  - Memorizzazione dei parametri in una tabella in memoria, e passaggio dell'indirizzo della tabella come parametro in un registro.
  - *Push* dei parametri nello stack da parte del programma. Il SO recupera i parametri con un *pop*.

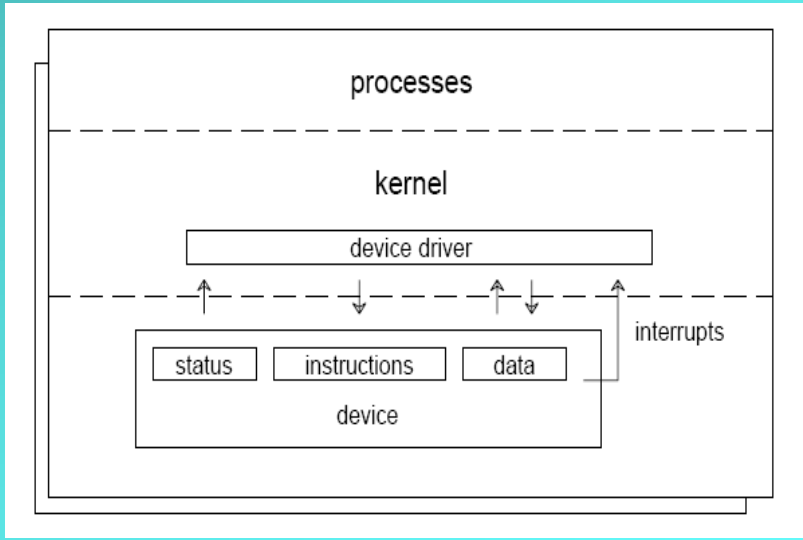
Esempio chiamate di sistema

cont= read(file, buffer, nbyte);



L’interfaccia dei dispositivi

- I dispositivi (devices) possono essere collegati alla CPU mediante vari meccanismi.
- Il più semplice è quello dei registri, che possono essere accessibili direttamente come zone di memoria o tramite comandi.
- I registri sono normalmente utilizzati per quattro scopi:
  - Trasferire informazioni di stato tra il dispositivo e la CPU
  - Trasferire istruzioni dalla CPU al dispositivo
  - Trasferire dati dal dispositivo alla CPU
  - Trasferire dati dalla CPU al dispositivo.



Polling & Interrupt

- Ci sono due modi in cui la CPU può sapere che un dispositivo ha completato la sua operazione:
  - **Polling**: interroga periodicamente il registro di stato del dispositivo finché non vede il bit Ready.
  - **Interrupt**: Richiede al dispositivo di generare un interrupt. I device driver nel kernel sono, in genere, concepiti per rispondere agli interrupt.

Evidentemente il polling è poco economico per la CPU che è continuamente impegnata a leggere il registro.

L’interprete dei comandi

- L’interprete dei comandi è un processo del S.O. che funge da interfaccia tra l’utente e il S.O. medesimo
- Attraverso l’interprete (shell) è possibile impartire comandi al S.O. e attivare dei processi.
- L’interprete può essere a linea di comando o grafico (come Windows). In questo secondo caso l’interfaccia è “a oggetti”: ciascun oggetto ha delle caratteristiche comuni e altre particolari.

Sistemi operativi Real-Time

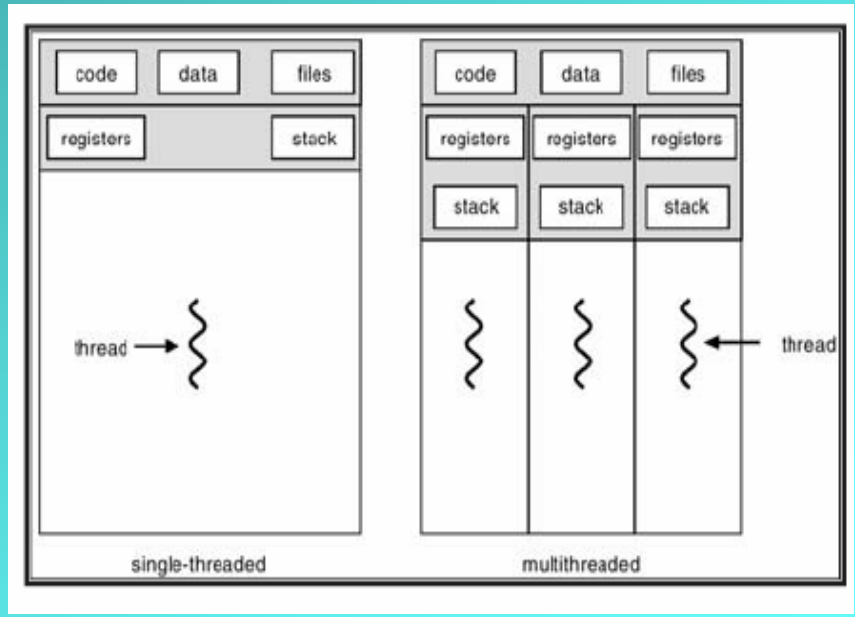
- Con il termine **tempo reale stretto** (*HRT-hard real time*) si intendono quei sistemi in grado di completare un'operazione critica in un tempo definito. Lettore di CD, pilota automatico, robot catena di montaggio,…
- **Prenotazione delle risorse**: un processo è accompagnato da una dichiarazione di tempo entro cui completare l'operazione; se è possibile lo scheduler accetta, altrimenti rifiuta la richiesta.
- Lo scheduler deve sapere quanto dura ciascuna operazione; questa richiesta non può essere garantita nei sistemi con memoria virtuale o con memoria secondaria.
- I sistemi HRT non possono offrire tutte le funzionalità dei moderni sistemi operativi.

Soft Real-Time

- Nelle elaborazioni in **tempo reale debole** (*SRT-soft real time*) ci si limita a richiedere che i processi critici abbiano una priorità maggiore dei processi ordinari.
- Il criterio di assegnazione delle risorse risulta **iniquo**, cioè ritarda l'esecuzione di alcuni processi, e può provocare situazioni di attesa indefinita.
- Tali sistemi sono d'uso generale e capaci di offrire, oltre alle funzioni tradizionali, un ambiente per l'esecuzione di applicazioni multimediali e per la grafica interattiva ad alte prestazioni.
- Il sistema deve disporre di uno scheduler per priorità, in cui la priorità dei processi in tempo reale non diminuisce col passare del tempo; l'allocatore deve avere una bassa latenza.

Processi Vs. Thread

- I thread sono simili ai processi, ma non hanno un blocco di contesto proprio.
- Di fatto sono dei sotto-processi che condividono il contesto e parte dei dati con il genitore.
- Possiedono gli stessi stati di un processo.

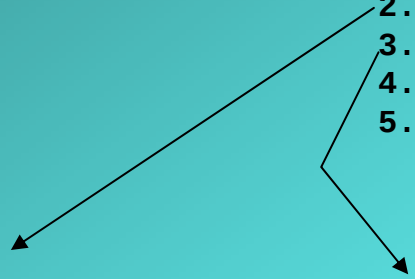


Vantaggio dei thread

- Somma dei numeri da 1 a N:

```
Singolo processo:
1.   Inizializza la variabile somma a 0
2.   Finché i<N
3.   somma=somma+I
4.   Incrementa I
5.   Torna a 2
6.   Stampa e esci
```

```
Due thread: Processo genitore
1.   Inizializza la variabile somma a 0
2.   Lancia processo somma pari
3.   Lancia processo somma dispari
4.   Aspetta la fine
5.   Stampa ed esci
```



```
Due thread: Thread pari
1.   Inizializza la variabile sum_p a 0
2.   Per i da 2 a N
3.   sum_p = sum_p+i
4.   i= i+2
5.   torna a 2
6.   somma= somma+sum_p
```

```
Due thread: Thread dispari
1.   Inizializza la variabile sum_d a 0
2.   Per i da 1 a N-1
3.   sum_d = sum_d+i
4.   i= i+2
5.   Torna a 2
6.   somma= somma+sum_d
```

Parallelismo =>Grande vantaggio
su architetture multiprocessore

Comunicazione inter-processo

- Come far mutuamente comunicare dati o informazioni di controllo fra i processi in esecuzione sull'elaboratore?
- Vari metodi:
  - Pipes
  - Segnali
  - semafori
  - Code di messaggi
  - Memoria condivisa
  - Socket
- Vanno collettivamente sotto il nome di InterProcess Communication (IPC).

Pipes & FIFOs

- Le pipes (tubi…) forniscono un flusso unidirezionale di informazione fra processi sullo stesso sistema
- L'informazione che passa in una pipe è transitoria, cioè, una volta letta non può essere ri-letta.
- Una serie di scritture ad un'estremità della pipe sarà riletta nel medesimo ordine all'altra estremità.
- Le FIFOs sono delle pipes dotate di un nome che non muoiono con il processo ma sono permanenti nel sistema fino ad un'esplicita cancellazione (come un file del S.O.).

Memoria condivisa

- Si tratta di regioni di memoria che possono essere accedute da più processi in lettura e scrittura
- Si rende necessario un meccanismo per regolare l'accesso concorrente alla memoria condivisa (vedi dopo).
- La versione con nome della memoria condivisa è la memoria mappata: il meccanismo associa ad un file del S.O. una regione di memoria virtuale ed i processi vi accedono come ad un file.

Code di messaggi

- Le code di messaggio sono un meccanismo che permette ai processi di scambiarsi informazioni in modo asincrono: il mittente non si aspetta conferma dal ricevente, il ricevente non si ferma in attesa del messaggio.
- Rispetto alle FIFO ci sono meccanismi che permettono di estrarre i messaggi dalla coda prima che raggiungano la fine.
- Le code di messaggio, una volta create, non muoiono con il processo che le ha create ma devono venire distrutte esplicitamente. Questo permette ad un processo di connettersi ad una coda già esistente.

Semafori

- I semafori sono un meccanismo che regola l'accesso alla memoria condivisa.
- Sono essenzialemnte dei contatori che possono essere aumentati, controllati o diminuiti.
- Esistono in due forme: binari o counting
  - Binari: consentono l'accesso se il loro valore è 0, lo bloccano in tutti gli altri casi. Il set li portano ad 1 il reset a 0.
  - Counting: vengono inizializzati ad un valore >=0, ogni volta che un processo li setta (take), diminuiscono di un unità, il reset (give - segnalazione) aumenta di un unità. L'accesso alla risorsa è consentito quando sono positivi, impedito quando negativi. In questo modo si contano i processi in attesa.