Obiettivi di un OS

***Convenienza** nell'uso del calcolatore rispetto ai potenziali utenti*

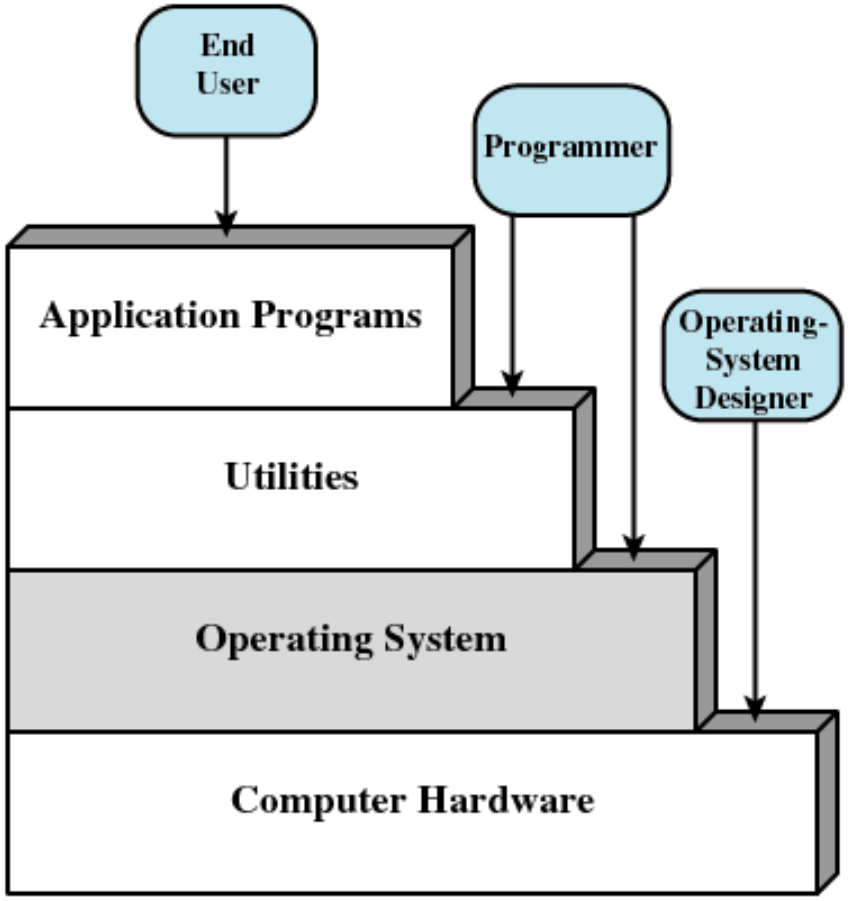
***Efficienza** nell'utilizzo del calcolatore e delle sue parti costitutive*

***Capacità di evolversi** rispetto a evoluzioni hardware, esigenze degli utenti e bachi*

Sistema Operativo come interfaccia (convenienza del SO)

Key Word: TRASPARENTE

Struttura Gerarchica di Hardware e Software



Il sistema operativo:

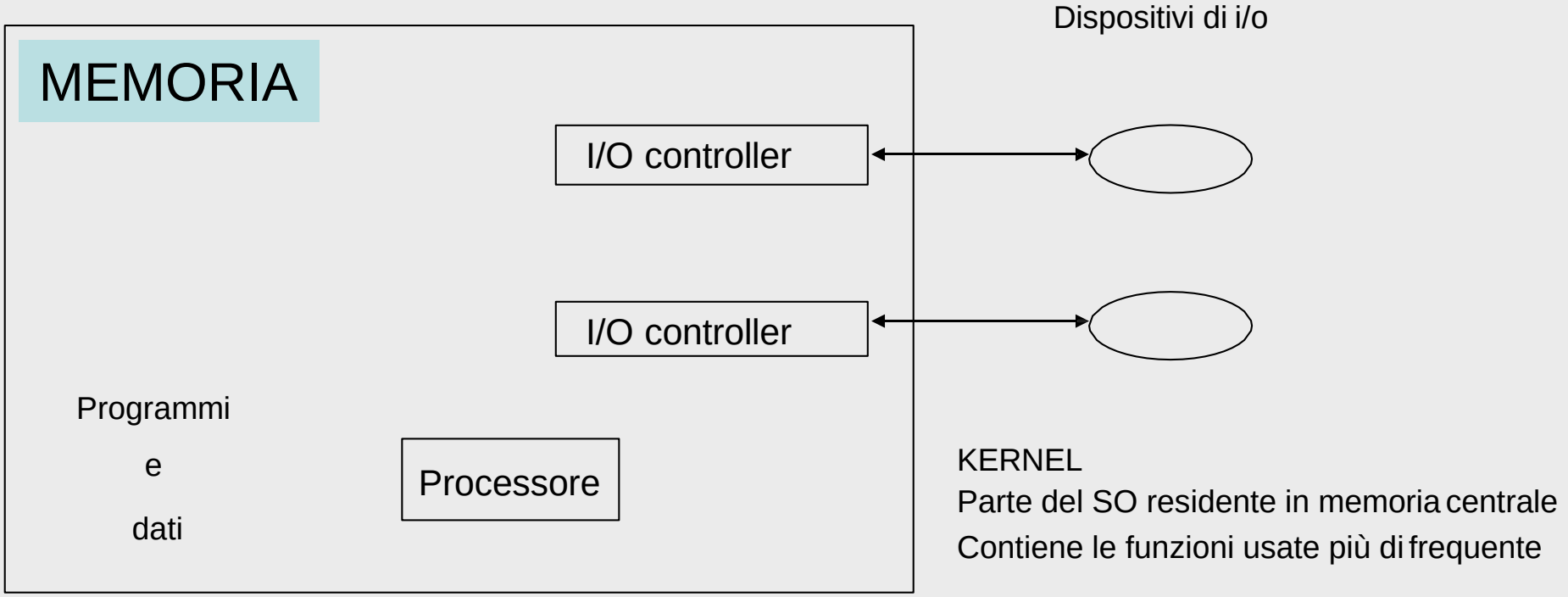
•nasconde i dettagli hardware al programmatore

•Fornisce una interfaccia per utilizzare il sistema

Servizi Offerti dal SO

- **Creazione dei programmi**: compilatore, debugger come utilità offerte al programmatore. Non sono parte del SO ma sono accessibili tramite esso
- **Esecuzione dei programmi**: caricamento in memoria dei programmi, inizializzazione dei dispositividi I/O, ecc.
- **Accesso ai dispositivi di I/O**: l'utente/programmatore ignora il set di istruzioni e i segnali dei dispositivi
- **Accesso controllato ai file**: comprensione del formato, meccanismi di protezione, associazione file indirizzi di memoria
- **Accesso al sistema** (inteso in senso lato)
- **Rilevazione e correzione degli errori** hardware o generati da programmi in esecuzione
- **Contabilità e statistiche d'uso** delle risorse, dei tempi di risposta (fine: migliorare le prestazioni)

SO come gestore delle risorse (efficienza del SO)



Il SO:

- *dirige la CPU nell'utilizzo delle altre risorse del sistema e nella temporizzazione dell'esecuzione dei programmi*
- *decide quando un programma in esecuzione può utilizzare una risorsa*
- *il processore stesso è una risorsa!!*

Batch Multi-Programmati

Mono-Programmazione

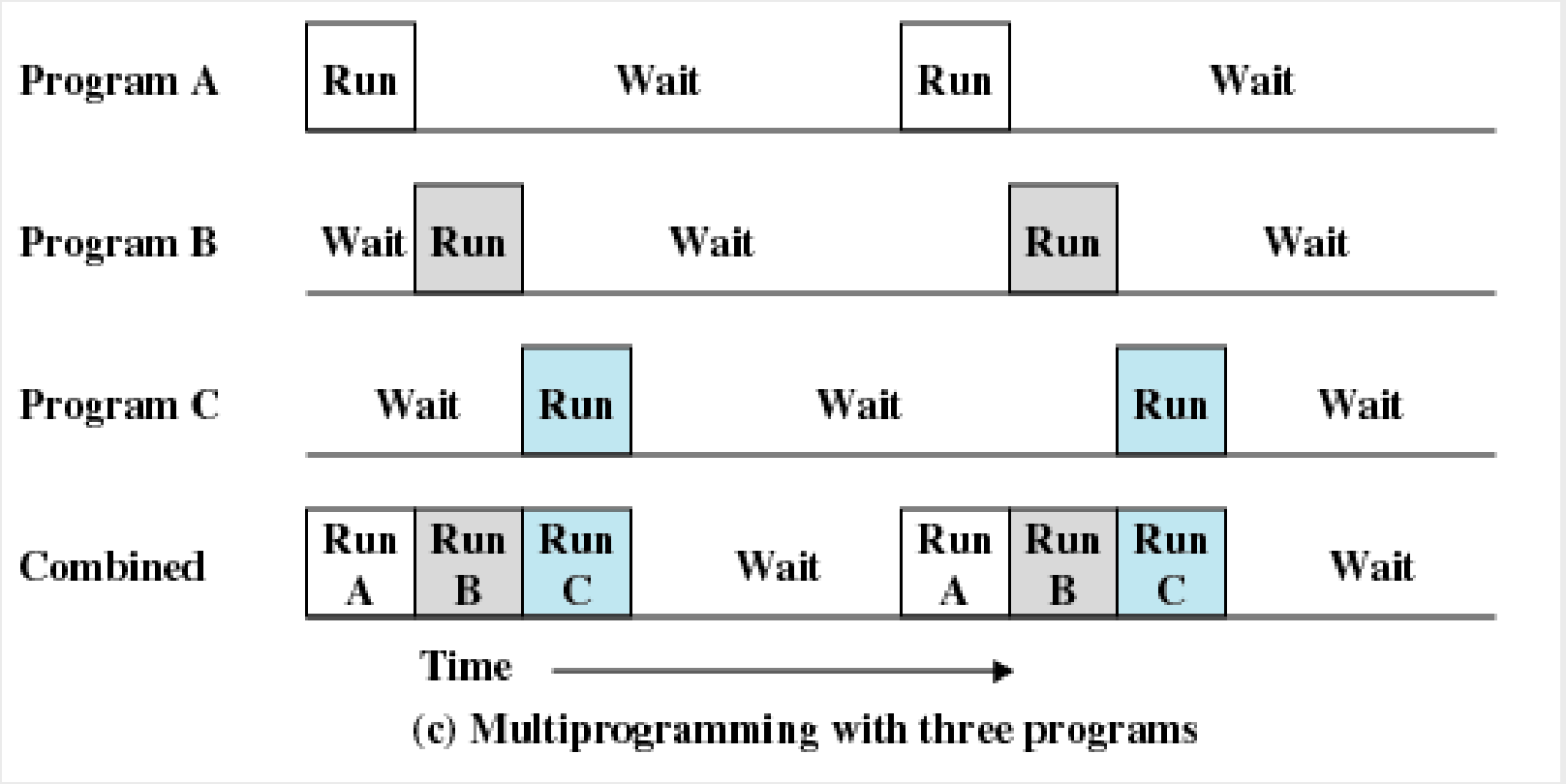
Lettura di un record	0.0015 sec.
Esecuzione di 100 istruzioni	0.0001 sec.
Scrittura di un record	0.0015 sec.
TOTALE	0.0031 sec.

Percentuale di utilizzo CPU

$$= \frac{0.0001}{0.0031} = 0.032 = 3.2\%$$

- Multi-programmazione:
  - Presenza di più programmi in memoria
  - Obiettivo: limitare l'inattività del processore, quando un job effettua una operazione di I/O la CPU può essere impegnata da un altro processo
  - Elaborazione seriale dei task

Multi-Programmazione
MULTI-TASKING



(c) Multiprogramming with three programs

Mono-Programmazione vs. Multi-Programmazione

	Job 1	Job 2	Job 3
Tipo	calcolo	I/O	I/O
Durata	5 min	15 min	10 min
Mem.	50 KB	100 KB	80 KB
Disco	No	No	Si
Termin.	No	Si	No
Stamp.	No	No	Si

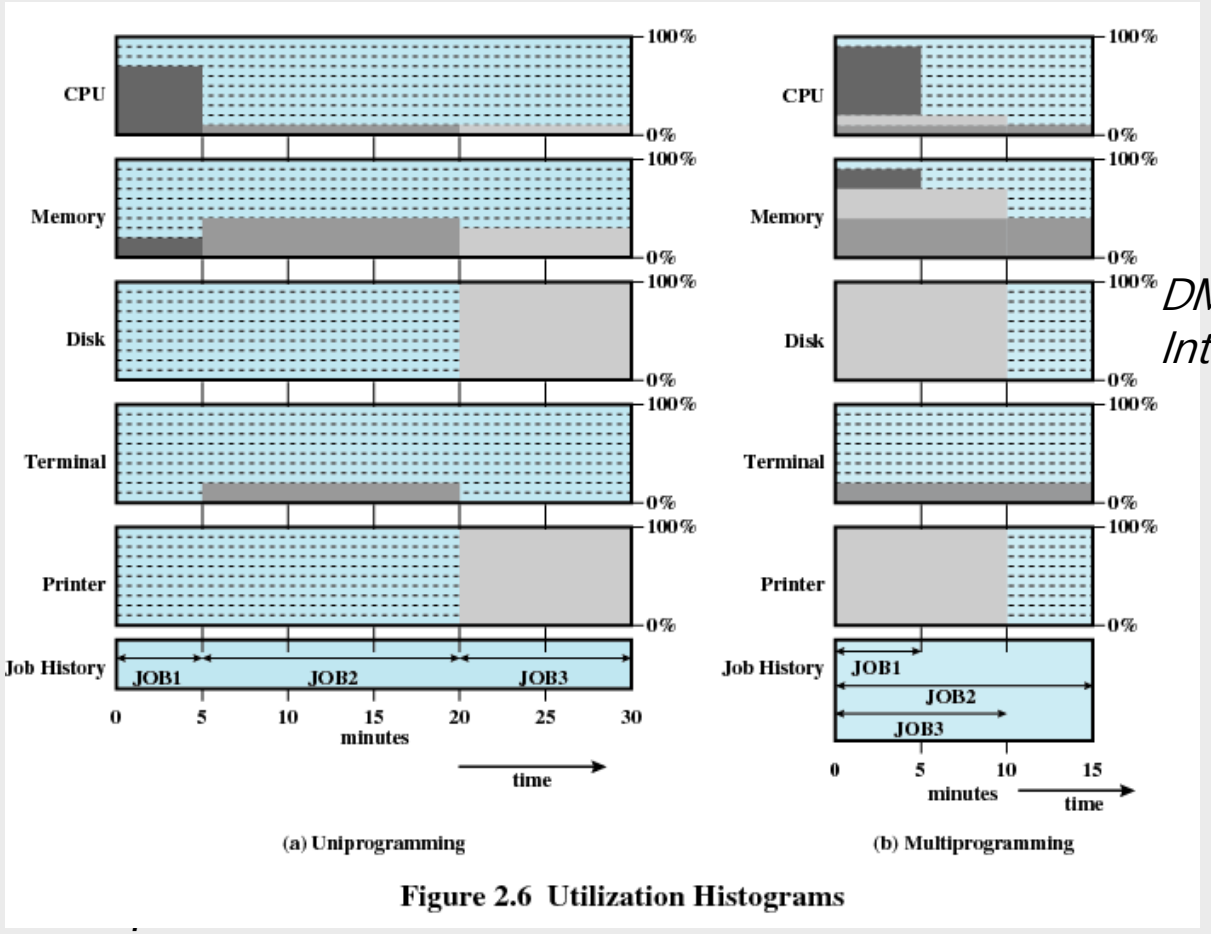


Figure 2.6 Utilization Histograms

DMA
Interrupt

Difficoltà della multiprogrammazione:
-Gestione della memoria
-Decidere quale job mandare in esecuzione (schedulazione)

Processo = Job = Task

- Un programma in esecuzione
- L'anima di un programma (!?)
- Una entità assegnata ad un processore e da essa eseguita

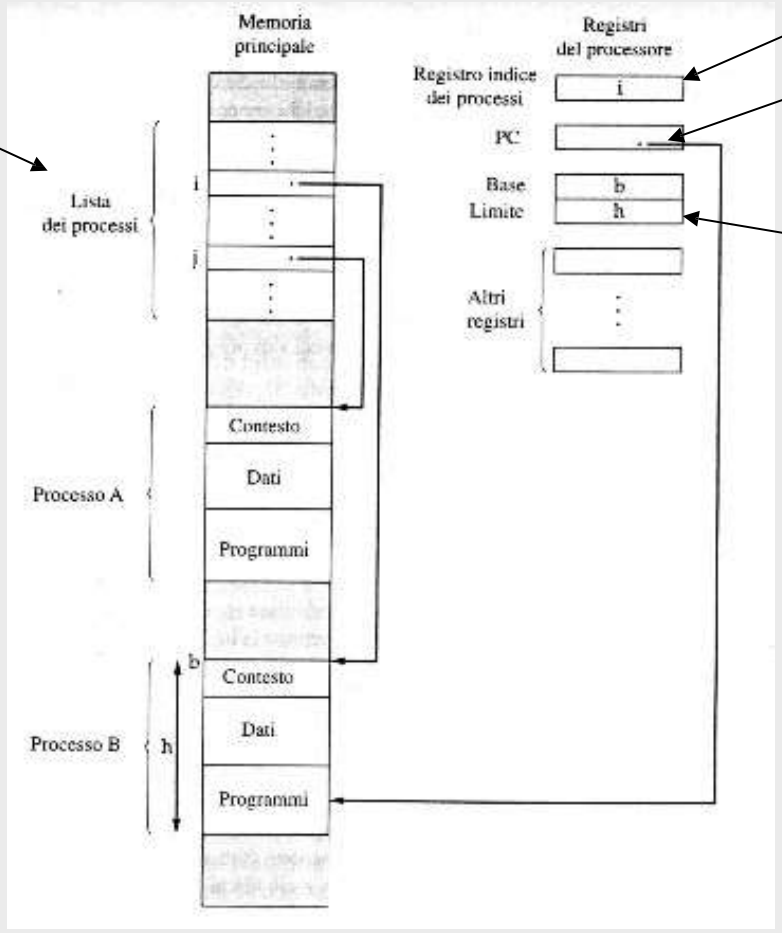
Componenti

- Programma
  - codice eseguibile
- Dati
  - variabili
  - spazio di lavoro
  - Buffer
- Contesto di esecuzione (info necessarie al SO per gestire il processo)
  - contenuto dei registri della CPU
  - Priorità,
  - Stato di esecuzione
  - Stato di attesa su un dispositivo di I/O

Implementazione di un processo

Gestita dal SO

```
struct
{programma,
 dati,
 contesto}
processo;
```



B è in esecuzione

Punta alla istruzione successiva del processo da eseguire

Protezione delle aree di memoria degli altri processi

Context Switching

Passaggio da un processo ad un altro:

1. *Salvataggio del primo contesto*
2. *Caricamento del secondo*

Gestione della Memoria

Il SO deve assolvere 5 compiti:

1. Isolamento dei processi
2. Allocazione e gestione automatica della memoria: la gerarchia delle memorie deve essere trasparente all'utente
3. Supporto alla programmazione modulare: variazione di dimensione dei programmi
4. Protezione e controllo dell'accesso: gestione di aree di memoria condivise tra i processi
5. Memorizzazione a lungo termine

Necessità soddisfatte da:

memoria virtuale: i programmi indirizzano la memoria con riferimenti logici ignorando gli aspetti fisici, quando un programma è in esecuzione solo una sua parte risiede effettivamente in memoria centrale

file system: implementa la memorizzazione a lungo termine

Schedulazione

La politica di allocazione delle risorse deve considerare i seguenti fattori:

- **Equità**: tutti i processi
  - appartenenti ad una stessa classe,
  - o con richieste simili,
  - o stesso costo,

  devono avere la stessa possibilità di accesso alla risorsa

- **Tempo di risposta differenziale**: il SO discrimina tra classi che hanno bisogno di risorse diverse e di tempi diversi
  - Es.: i processi con forte uso di I/O vengono schedulati per primi

- **Efficienza**: massimizzare il throughput, minimizzare il tempo di risposta