

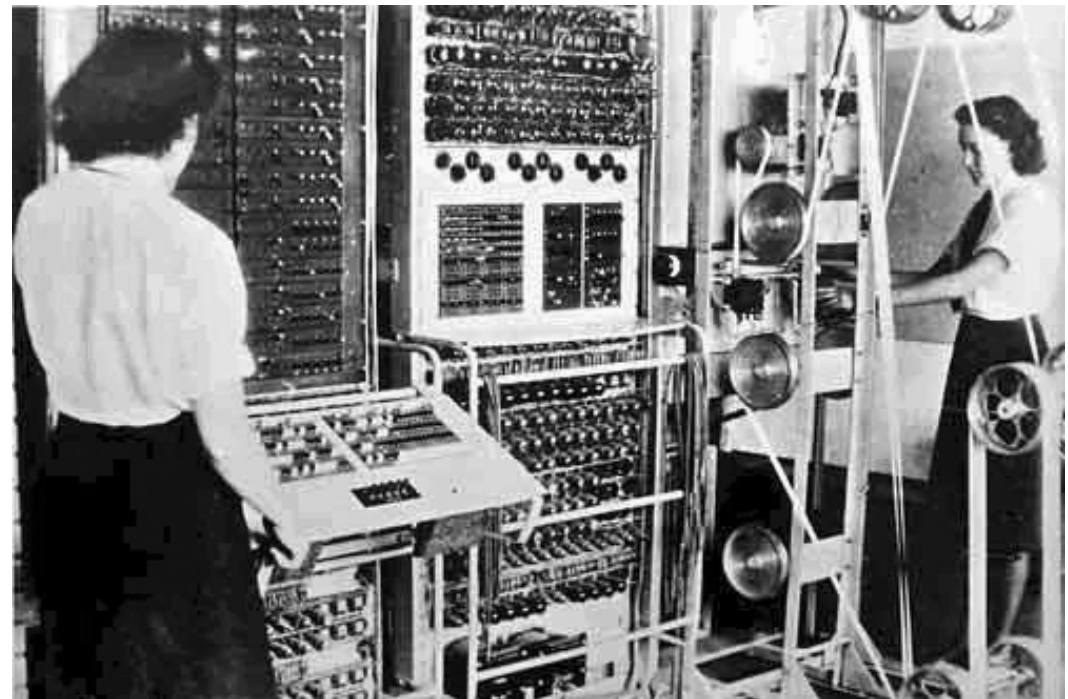
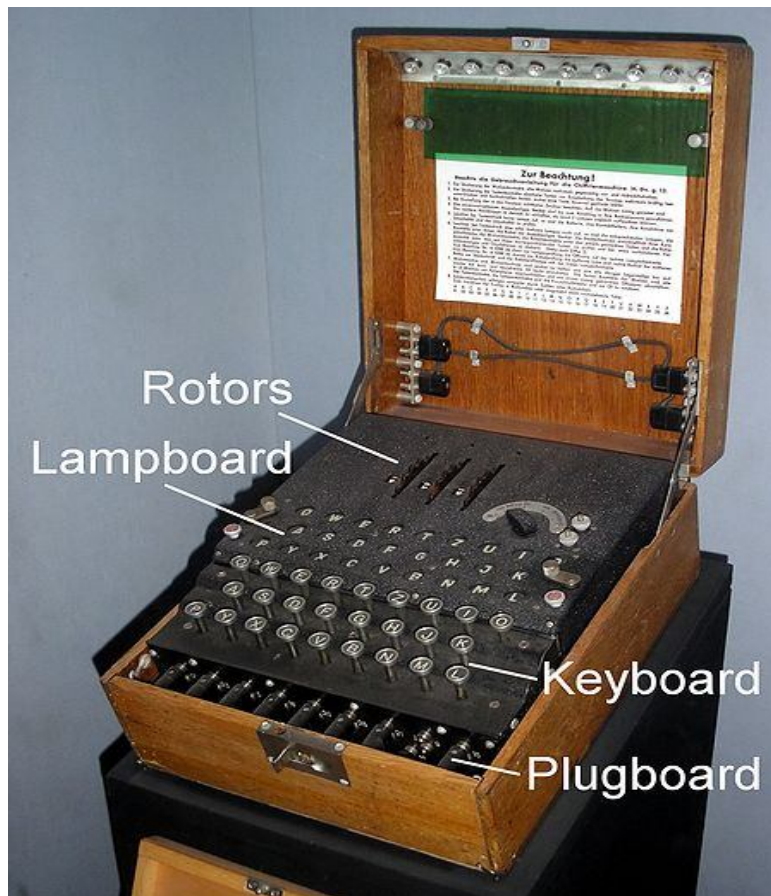
Sistemi Operativi

Introduzione

Docente: Claudio E. Palazzi
cpalazzi@math.unipd.it

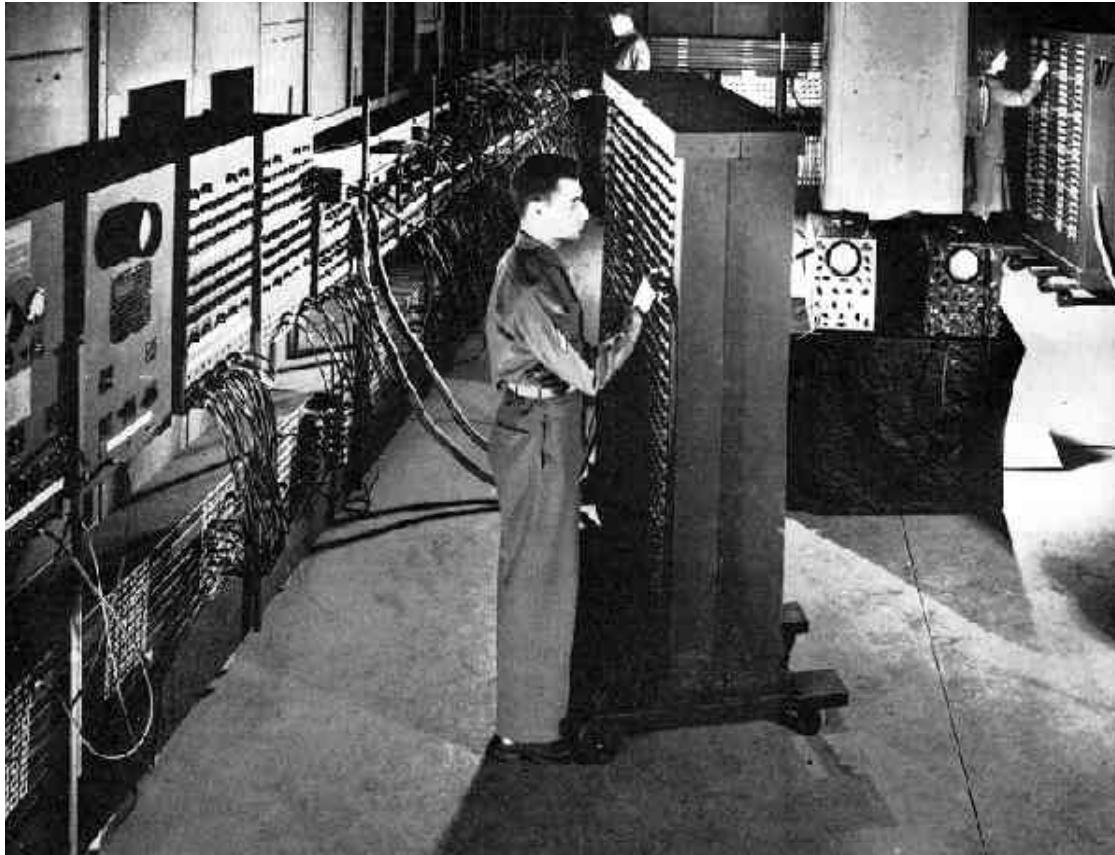
Crediti per queste slides al Prof. Tullio Vardanega

Cenni storici – 0



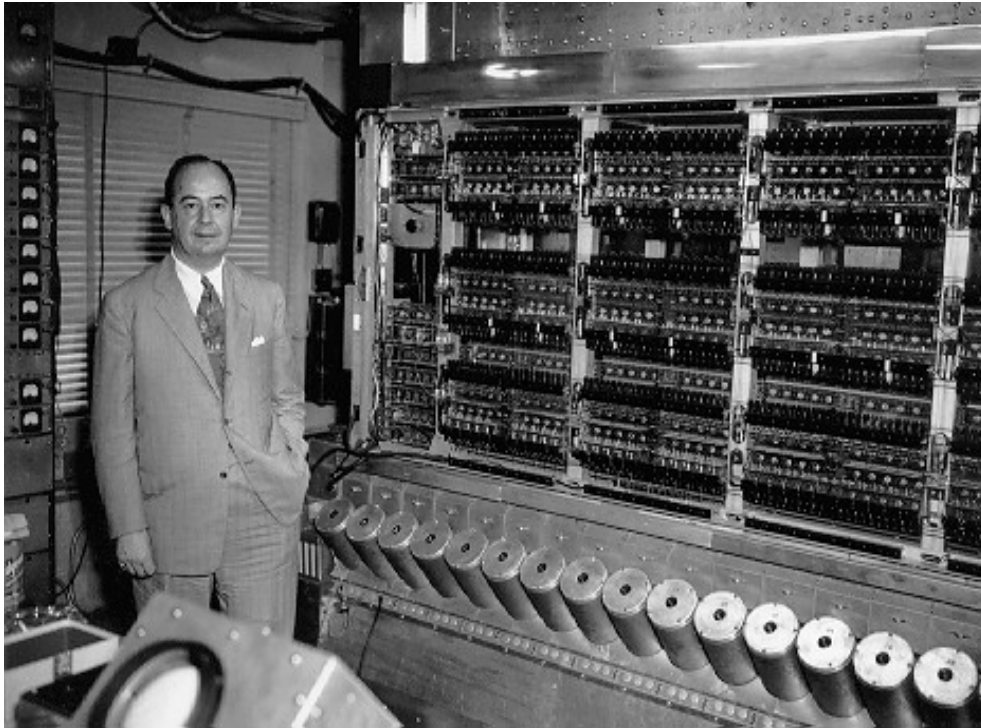
- **1944:**
Enigma vs Colossus (Mark I/II)

Cenni storici – 0 bis



- **1947: ENIAC**
 - Creato per velocizzare calcolo tabelle balistiche
 - Electronic Numerical Integrator And Calculator
 - I “Computer” erano gli operatori
 - Per programmare il calcolo della circonferenza configurare serie di cavi e 3000 switch
 - Bit: Binary Digit
 - Basato su sistema decimale
 - Prima accensione causò blackout
 - Unità di Controllo: oscilloscopio

Cenni storici – 0 bis



- **1949: EDVAC**

- Electronic Discrete Variable Automatic Computer
- Transistor (abbatte consumo energetico)
- Architettura di von Neumann
- Programma Memorizzato
- Basato su sistema binario
- Costo 500.000 \$
- Lettore schede perforate (1953)
- Unità virgola mobile (1958)
- Peso di quasi 8 tonnellate
- 30 operatori (cambio ogni 8 ore)

Cenni storici – 1

- **Anni '50:** i S/O hanno origine con i primi elaboratori a programma memorizzato
 - Modalità di esecuzione *batch* (a lotti) gestita da un operatore umano
 - **Tutto** l'elaboratore a disposizione di **un solo** programma per tutto il tempo della sua esecuzione
 - Immissione di un programma mediante interruttori binari frontali (switch) o schede perforate
 - Emissione dei risultati mediante lampadine, testo stampato, schede perforate
 - I primi videogiochi:
 - OXO su EDVAC
 - Tennis for Two su oscilloscopio



Cenni storici – 2

- A partire dagli **anni '60**
 - Nuovi compilatori, nuovi linguaggi di programmazione (es. BASIC, per gli studenti), nuovi strumenti di sviluppo
- Ancora gestione a lotti
 - Immissione ed emissione di programmi e dati ancora molto laboriose (= molto costose e molto lente)
 - L'operatore umano è ancora necessario per eseguire le operazioni di ingresso / uscita
- Serie PDP
 - PDP-8 venduti 55.000 mila esemplari
 - Primo computer di successo commerciale



Cenni storici – 3

- L'esecuzione di più **lavori** in modalità a lotti può essere facilmente gestita da un S/O **residente**
 - Più caricamenti seguiti da una fase ininterrotta di lavoro e dal ritrovamento dei rispettivi risultati
 - Ordine di esecuzione **predeterminato**
 - Quello di caricamento
 - Secondo il livello di privilegio del richiedente
- Operazioni di I/O fino a 1.000 volte più lente dell'elaborazione
 - Esecuzione *off-line* (es. Stampante non collegata al computer)
 - Senza richiedere tempo di elaboratore
 - Sovrapposizione tra I/O ed elaborazione

Cenni storici – 4

- Sovrapposizione sempre più conveniente al crescere delle prestazioni dei dispositivi
 - Tecnica detta di *spooling* (*Simultaneous Peripheral Operation On-Line*)
 - Si effettua *spooling* quando l'emissione o la ricezione di dati avviene in parallelo all'esecuzione di altri lavori
 - Esempi
 - Invio di una richiesta di stampa
 - Caricamento di un programma
 - Invio di un messaggio di posta elettronica
 - **Senza interrompere** il lavoro in corso

Cenni storici – 5

- **Multiprogrammazione**

- Desiderabile poter eseguire diversi lavori simultaneamente
 - In ambito mono-processore il parallelismo è solo simulato
- Occorre controllare l'assegnazione dell'accesso alle risorse della macchina
 - Esempio: per **quanti di tempo** in modalità *time-sharing* sotto il controllo del S/O

Cenni storici – 6

- IBM fondata nel 1968
- Prima installazione commerciale di UNIX nel 1972
- Microsoft fondata nel 1975
- Apple fondata nel 1977



IBM portable PC



Atari 400



Apple II

Cenni storici – 7

- Anni '80
 - Sinclair ZX 80 venduto a soli 100\$
 - Floppy 3.5 e CD
 - Protocolli TCP/IP per Internet
 - Diffusione PC e Console di gioco
 - Meno di 1 MB di memoria
 - CPU nell'ordine del 10MHz
 - Plug & Play e Multitasking Preemptive (su Commodore Amiga 500)
 - Su PC solo con Windows 95 (11 anni dopo...)



Cenni storici – 8

- Anni '90
 - Microsoft Office
 - World Wide Web
 - Applicazioni distribuite (anche giochi)
 - Centinaia di milioni di PC venduti all'anno
- Anni 2000
 - Più laptop venduti che desktop
 - Maggioranza Windows (XP, Vista, 7)
- Anni 2010
 - Più smartphone e tablet venduti che laptop/desktop
 - Netbook
 - Cloud
 - Sistema Operativo? (quale e dove?)

Definizione di S/O

- Un insieme di utilità progettate per...
 1. Offrire all'utente un'astrazione più semplice e potente della macchina *assembler*
 - Concetto di “**macchina virtuale**”
 - ambiente virtuale dove eseguire applicazioni
 - originariamente per sistemi multi-utente
 - Più semplice da usare (es., senza bisogno di conoscenze di microprogrammazione 😊)
 - Più potente (es., usando la memoria secondaria per realizzare una più ampia memoria principale virtuale)
 2. Gestire in maniera **ottimale** le risorse fisiche e logiche dell'elaboratore
 - Ottimalità è la minimizzazione dei tempi di attesa e la massimizzazione dei lavori svolti per unità di tempo

Nozione di processo

- Un processo è un **programma in esecuzione** e corrisponde a
 1. L'insieme **ordinato** di **stati** assunti dal programma nel corso dell'esecuzione (sulla sua macchina virtuale)
 - Processo come “automa a stati”
 2. L'insieme **ordinato** delle **azioni** effettuate dal programma nel corso dell'esecuzione (sulla sua macchina virtuale)
 - Processo come “attore” (operatore di azioni)

Realizzazione di processo

- Spazio di **indirizzamento logico**
 - La memoria della **macchina virtuale** che il processo può leggere e scrivere
 - Memoria virtuale organizzata a pagine e/o segmenti
 - Programma eseguibile
 - Dati del programma
 - Organizzazione dell'informazione in forma di *file*
 - Aree di lavoro

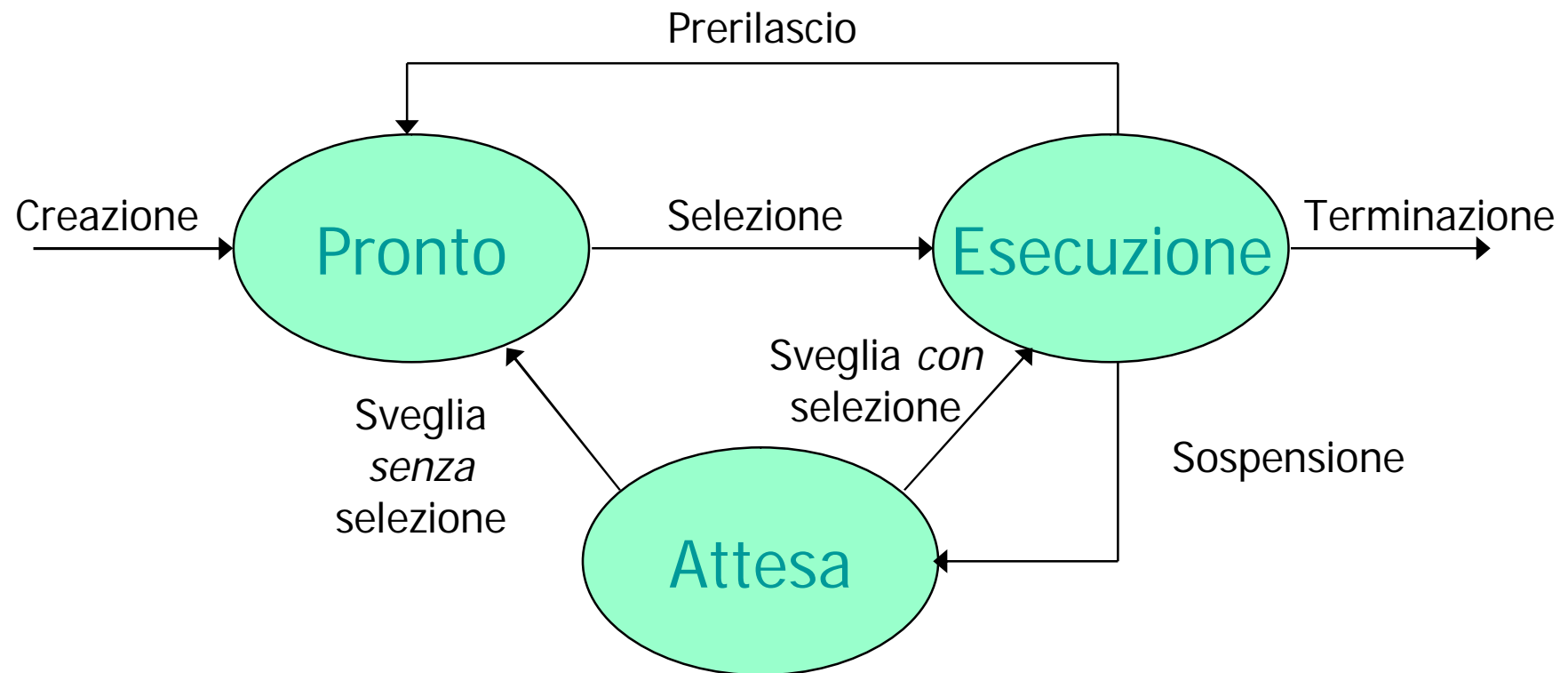
Caratteristiche di processi – 1

- In un sistema coesistono processi utente e di S/O
 - Possono cooperare tra loro ma hanno privilegi diversi
- I processi avanzano concorrentemente
 - Il S/O assegna loro le risorse necessarie secondo diverse politiche di ordinamento
 - A divisione di tempo
 - A livello di priorità (urgenza)
- I processi possono dover comunicare e sincronizzarsi tra loro
 - Il S/O deve fornire i meccanismi e i servizi necessari

Caratteristiche di processi – 2

- Un processo può creare processi “figli”
 - Esempio
 - Un processo interprete di comandi (*shell*) lancia un processo figlio per eseguire un comando di utente
- I processi vengono
 - **Creati** per eseguire un lavoro
 - **Sospesi** per consentire l'esecuzione di altri processi
 - **Terminati** al compimento del lavoro assegnato
 - Un processo figlio che sopravvive alla terminazione del processo padre è detto “orfano” e può essere molto dannoso

Stati di avanzamento di processo



Gestore dei processi – 1

- Costituisce il cuore o nucleo del S/O (*kernel*)
 - Gestisce ed assicura l'avanzamento dei processi
 - Stato di avanzamento
 - In esecuzione, pronto per l'esecuzione, sospeso in attesa di un evento (una comunicazione, la disponibilità di una risorsa, ...)
 - La scelta del processo da eseguire ad un dato istante si chiama ordinamento (*scheduling*)
 - Il gestore decide il cambio di stato dei processi
 - Per divisione di tempo
 - Per trattamento di eventi (es., risorsa libera / occupata)

Gestore dei processi – 2

- Compiti del nucleo di S/O
 - Gestire l'avanzamento dei processi
 - Registrando ogni transizione nel loro stato di attivazione
 - Gestire le interruzioni esterne (all'esecuzione corrente) causate da
 - Eventi di I/O
 - Situazioni anomale rilevate da altri processi o componenti del S/O
 - Consentire ai processi di accedere a risorse di sistema e di attendere eventi

Gestore dei processi – 3

- La politica di ordinamento deve essere equa (*fair* → *fairness*)
 - Processi pronti per eseguire devono avere l'opportunità di farlo
 - Processi in attesa di risorse devono avere l'opportunità di accederle
- I meccanismi e servizi di comunicazione e sincronizzazione devono essere efficaci
 - Il dato (o segnale) inviato da un processo mittente deve raggiungere il destinatario in un tempo breve e in modo sicuro

Definizione di risorsa

- **Risorsa** è qualsiasi elemento fisico (*hardware*) o logico (realizzato a *software*) necessario alla creazione, esecuzione e avanzamento di processi
- Le risorse possono essere
 - Durevoli (es., CPU)
 - Consumabili (es., memoria fisica)
 - Ad accesso divisibile o indivisibile
 - Divisibile se tollera alternanza con accessi di altri processi
 - Indivisibili se *non* tollera alternanza durante l'uso
 - Ad accesso individuale o molteplice
 - Molteplicità fisica o logica (virtualizzata)

Risorsa CPU

- Risorsa indispensabile per l'avanzamento di tutti i processi
- A livello fisico (*hardware*) corrisponde alla CPU
- A livello logico (sotto gestione *software*) può essere vista come una **macchina virtuale**
 - Offerta dal S/O alle sue applicazioni

Risorsa memoria

- Scrittura: risorsa ad accesso individuale
- Lettura: risorsa ad accesso multiplo
- La gestione *software* la **virtualizza** (usandola insieme alla memoria secondaria) attribuendone l'accesso ai vari processi secondo particolari politiche
- Se virtualizzata, diventa riutilizzabile e prerilasciabile
 - Altrimenti consumabile e indivisibile
 - Gestione velocizzata con l'utilizzo di supporto *hardware*

Risorsa I/O

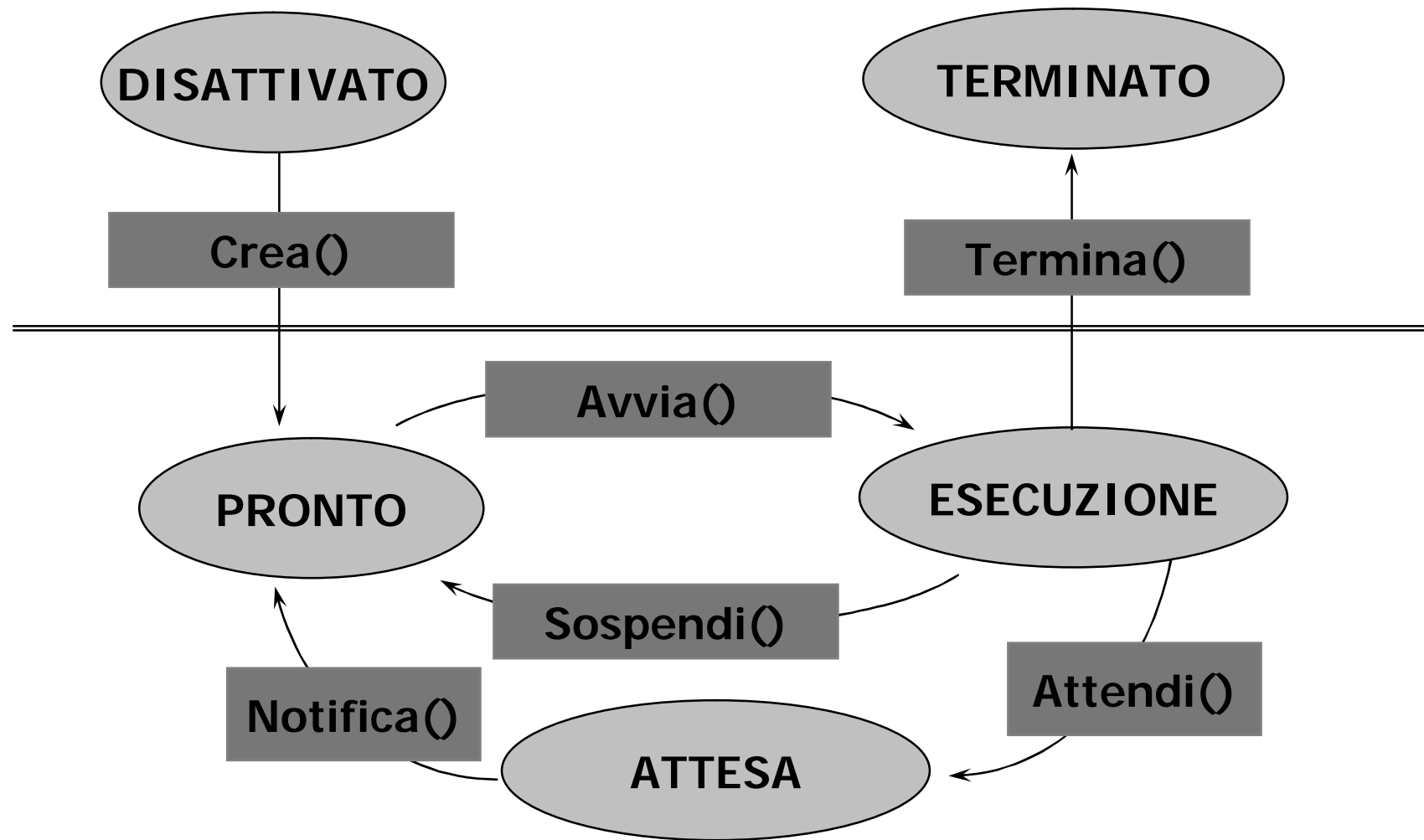
- Risorse generalmente riutilizzabili, non prerilasciabili, ad accesso individuale
- La gestione *software* ne facilita l'impiego nascondendone le caratteristiche *hardware* e uniformandone il trattamento
- L'accesso fisico ha bisogno di utilizzare programmi proprietari e specifici
 - *BIOS*

Caricamento del S/O

Il S/O può risiedere

- Permanentemente in ROM
 - Soluzione tipica di sistemi di controllo industriale e di **sistemi dedicati**
- In memoria secondaria per essere caricato (tutto o in parte) in RAM all'attivazione di sistema (*bootstrap*)
 - Adatto a sistemi di elevata complessità oppure predisposti al controllo (alternativo) da parte di più istanze di S/O
 - In ROM risiede solo il caricatore di sistema (*bootstrap loader*)

Stati di avanzamento di processo 2



Stati – 1

DISATTIVATO

Il programma è in memoria secondaria. Un supervisore lo carica in memoria mediante una chiamata di sistema che crea una struttura di controllo di processo (*Process Control Block, PCB*)

PRONTO

Il processo, pronto per l'esecuzione, rimane in attesa del suo turno

ESECUZIONE

Il processore è stato attribuito al processo selezionato, la cui esecuzione avanza

Stati – 2

ATTESA

Il processo è sospeso in attesa di una risorsa attualmente non disponibile o di un evento non ancora verificatosi

TERMINATO

Il processo ha concluso regolarmente le sue operazioni e si predispone ad abbandonare la sua **macchina virtuale**

Transizioni – 1

Crea()

Assegna una macchina virtuale a un nuovo processo, aggiornando la lista dei processi pronti (*ready list*)

Avvia()

Manda in esecuzione il primo processo della lista dei pronti

Sospendi()

Il processo in esecuzione ha esaurito il suo quanto di tempo e torna in fondo alla lista dei pronti

Transizioni – 2

- **Attendi()**
 - Il processo richiede l'uso di una risorsa o l'arrivo di un evento e viene sospeso se la risorsa è occupata o se l'evento non si è ancora verificato
- **Notifica()**
 - La risorsa richiesta dal processo bloccato è di nuovo libera o l'evento atteso si è verificato. Il processo ritorna nella lista dei pronti
- **Termina()**
 - Il processo in esecuzione termina il suo lavoro e rilascia la **macchina virtuale**

Strutture di rappresentazione

- Modello di processo realizzato tramite struttura a tabella (**Process Table**)
 - Array di strutture
- Ogni processo è rappresentato da un **descrittore** (**Process Control Block**) contenente
 - Identificatore del processo
 - Contesto di esecuzione (stato interno) del processo
 - Stato di avanzamento del processo
 - Priorità (iniziale ed attuale)
 - Diritti di accesso alle risorse e privilegi
 - Puntatore al PCB del processo padre e degli eventuali processi figli
 - Puntatore alla lista delle risorse assegnate alla macchina virtuale del processo
 - ... Vedi Fig. 2.4 nel libro

Campi di una Process Table

Process management	Memory management	File management
Registers Program counter Program status word Stack pointer Process state Priority Scheduling parameters Process ID Parent process Process group Signals Time when process started CPU time used Children's CPU time Time of next alarm	Pointer to text segment Pointer to data segment Pointer to stack segment	Root directory Working directory File descriptors User ID Group ID

Passi principali seguenti interrupt

1. Hardware stacks program counter, etc.
2. Hardware loads new program counter from interrupt vector.
3. Assembly language procedure saves registers.
4. Assembly language procedure sets up new stack.
5. C interrupt service runs (typically reads and buffers input).
6. Scheduler decides which process is to run next.
7. C procedure returns to the assembly code.
8. Assembly language procedure starts up new current process.

Ordinamento di processi

- Diversi metodi utili per determinare quando porre un processo in stato di esecuzione in sostituzione di un altro (**switch**)
 - **Scambio cooperativo** (*cooperative* o *non pre-emptive switching*)
 - Il processo in esecuzione decide quando passare il controllo al processo successivo
 - Windows 3.1 ☹
 - **Scambio a prerilascio**: il processo in esecuzione viene rimpiazzata da
 - Un processo pronto a priorità maggiore (*priority-based pre-emptive switching*) → Sistemi detti “a tempo reale”
 - All'esaurimento del suo quanto di tempo (*time-sharing pre-emptive switching*) → Unix, Windows NT (misti)

Dispatcher – 1

- Il componente che avvia processi all'esecuzione (ma non il selettore, *scheduler*!) viene detto *dispatcher*
 - Deve essere molto efficiente perché gestisce *ogni* scambio
 - Deve salvare il contesto del processo in uscita, installare quello del processo in entrata (*context switch*) e affidargli il controllo della CPU
- L'efficienza del *dispatcher* si misura in
 - Percentuale di utilizzo della CPU
 - Numero di processi avviati all'esecuzione per unità di tempo
 - Durata di permanenza di un processo in stato di pronto

Dispatcher – 2

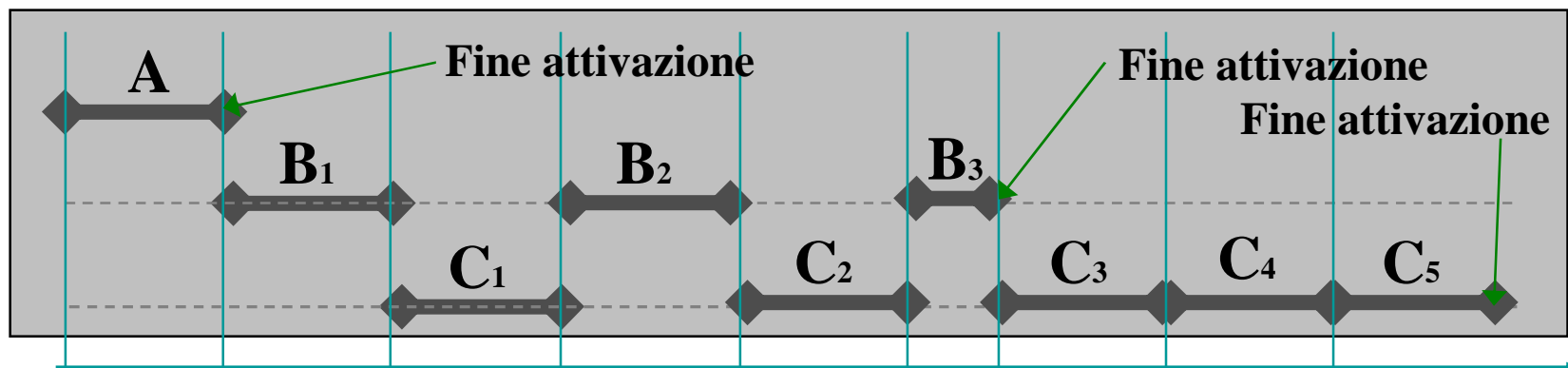
- I processi in stato di pronto sono accodati in una struttura detta **lista dei pronti** (*ready list*)
- La più semplice gestione della lista è con tecnica a coda (*First-Come-First-Served, FCFS*)
 - Il primo processo ad entrare in coda sarà anche il primo avviato all'esecuzione
 - Facile da realizzare e da gestire
 - La garanzia di esecuzione di altri processi (*fairness*) dipende dalla politica di scambio
 - Lo scambio cooperativo non offre garanzie

Dispatcher – 3

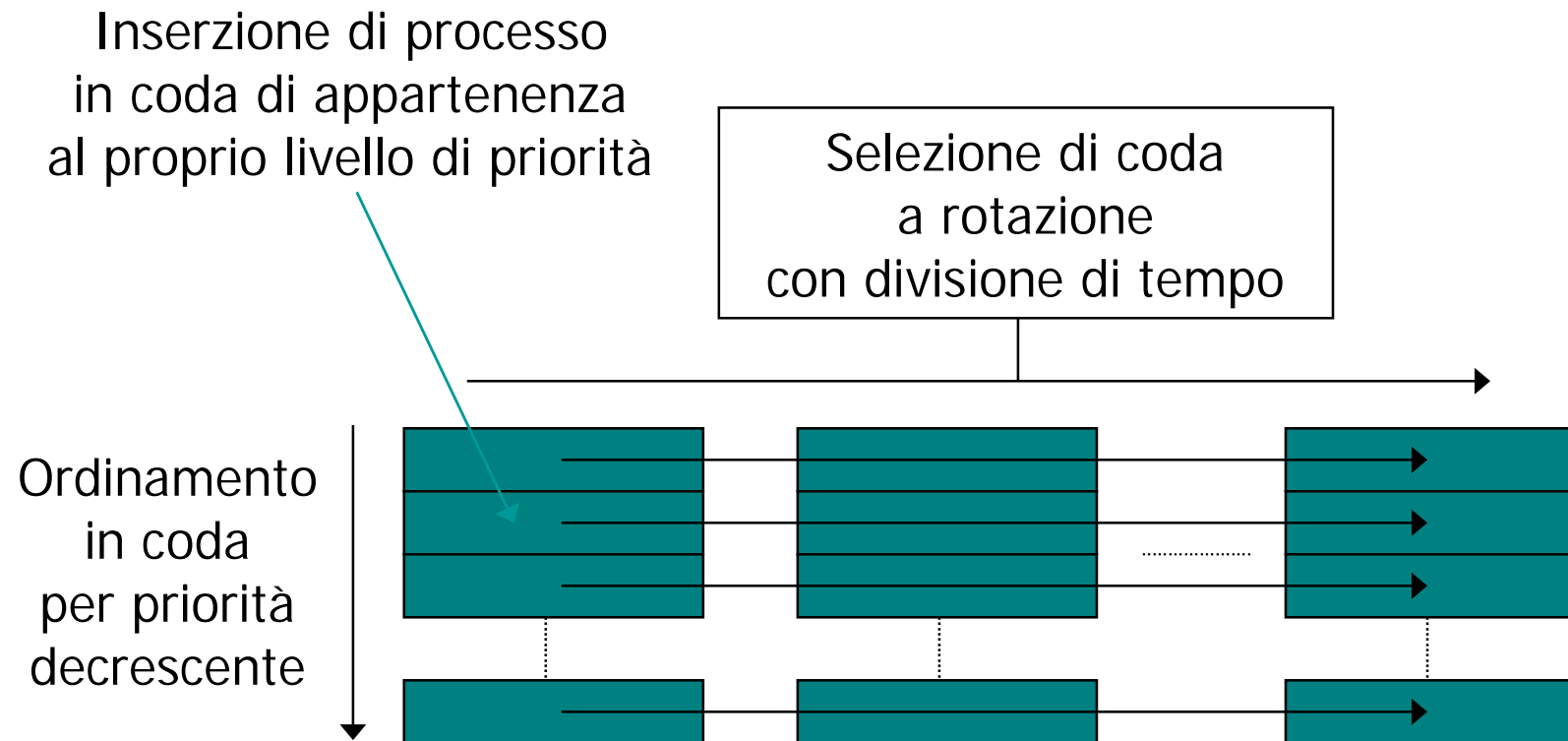
- Le attività di un processo tipicamente comprendono sequenze di azioni eseguibili dalla CPU intervallate da sequenze di azioni di I/O
- I processi si possono dunque classificare in
 - *CPU-bound*
 - Comprendenti attività sulla CPU e di durata *molto lunga*
 - *I/O-bound*
 - Comprendenti attività di breve durata sulla CPU, intervallate da attività di I/O molto lunghe
- La tecnica FCFS penalizza i processi della classe *I/O-bound*

Dispatcher – 4

- Imponendo la suddivisione di tempo (*time-sharing*) sulla politica *FCFS* si deriva una tecnica di rotazione detta *round-robin*
- Vediamone l'applicazione su tre processi A, B e C con tempi di esecuzione 2, 5 e 10 ms e quanto di tempo 2 ms



Politica a rotazione con priorità



Dispatcher – 5

- A ogni singolo processo possiamo attribuire una priorità individuale che denota il suo livello di privilegio nel sistema
- Processi diversi possono poi essere categorizzati per attributi (p.es., *CPU-bound*, *I/O-bound*)
- Possiamo allora istituire una coda per ciascuna categoria di processo e ordinarla a priorità
- Stabiliamo poi una politica di ordinamento tra code (p.es.: *round-robin*)
- Otteniamo una politica di ordinamento a livelli
 - Rotazione tra code e con priorità entro ciascuna coda

Dispatcher – 6

- Possiamo anche facilmente (e utilmente) definire una politica **duale** alla precedente
 - Istituiamo una coda per ogni livello di priorità attribuita ai processi
 - Selezioniamo la coda a priorità più elevata
 - Applichiamo la politica a rotazione (*round-robin*) sul processo selezionato
 - Otteniamo la politica a priorità con rotazione
 - Selezione prioritaria tra code e a rotazione equa entro ciascuna coda

Politica a priorità con rotazione

