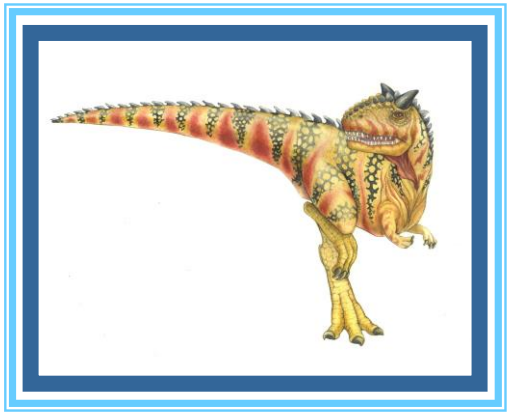
Introduzione

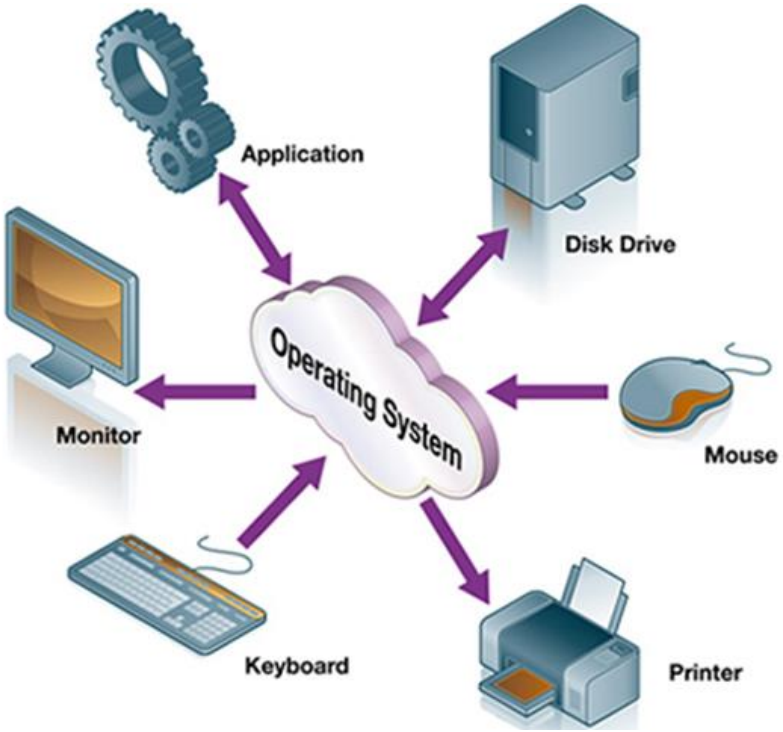


Obiettivi

- Descrivere l'organizzazione generale di un sistema di calcolo con particolare riferimento alle architetture multiprocessore
- Analizzare i più importanti componenti di un sistema operativo e il ruolo degli interrupt
- Illustrare la transizione dalla modalità utente alla modalità kernel
- Discutere l'utilizzo dei sistemi operativi in ambienti di calcolo ampiamente differenziati
- Fornire esempi di sistemi operativi free e open–source

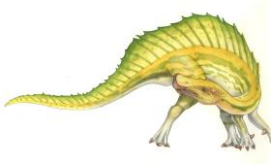
Sommario

- Che cos'è un sistema operativo
- Organizzazione del sistema di calcolo
- Architettura degli elaboratori
- Struttura del sistema operativo
- Attività del sistema operativo
  - Gestione dei processi
  - Gestione della memoria
  - Gestione dei file
  - Gestione dei dispositivi di I/O
- Protezione e sicurezza
- Virtualizzazione
- Strutture dati del kernel
- Ambienti di calcolo
- Sistemi operativi open–source



Che cos'è un sistema operativo? – 1

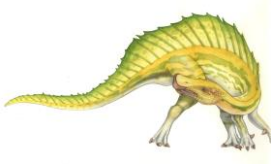
- Il Sistema Operativo perfetto è invisibile all'utente
- Ogni utente pensa in modo diverso all'utilizzo del computer: ha una propria *visione astratta* del sistema di elaborazione, che tiene conto solo di quelle caratteristiche che l'utente ritiene importanti
- Il sistema operativo (SO) è un software che agisce da intermediario tra l'utente e l'hardware del computer
  - Deve fornire servizi e caratteristiche presenti nelle visioni astratte di *tutti* i suoi utenti — servizi che mantiene aggiornati, evolvendosi nel tempo per adeguarsi alle loro esigenze



Che cos'è un sistema operativo? – 2

Che cos'è un sistema operativo? – 3

- Scopi del sistema operativo
  - Eseguire i programmi, supportando l'utente nel "risolvere problemi"
  - Rendere agevole l'interfaccia fra l'uomo e la macchina — il sistema di calcolo è "conveniente" da usare
  - Gestire in modo efficiente le risorse (hardware/software) del sistema di calcolo
  - Prevenire le interferenze fra/con le attività degli utenti
  - Dare agli utenti e alle applicazioni una visione "astratta" del sistema di calcolo



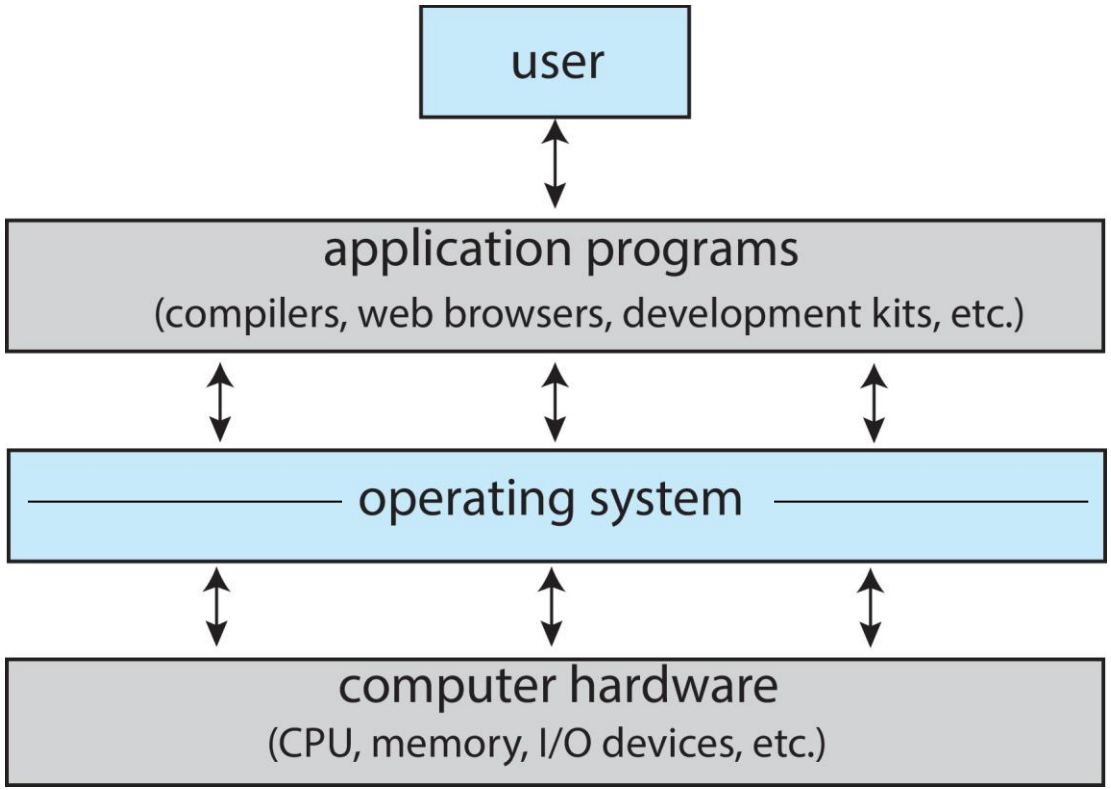
Organizzazione del sistema di calcolo

- Il sistema di calcolo si suddivide in quattro componenti principali:
  - Hardware — fornisce le risorse fondamentali di calcolo
    - CPU, GPU, memorie, device di I/O
  - Sistema operativo
    - Controlla e coordina l'utilizzo delle risorse hardware da parte dei diversi utenti e delle varie applicazioni
  - Programmi applicativi — definiscono le modalità di utilizzo delle risorse del sistema, per risolvere i problemi di calcolo degli utenti
    - Word processor, compilatori, Web browser, sistemi per basi di dati, fogli di calcolo, giochi
  - Utenti
    - Persone, altri macchinari, altri elaboratori

Le componenti del sistema di calcolo – 1

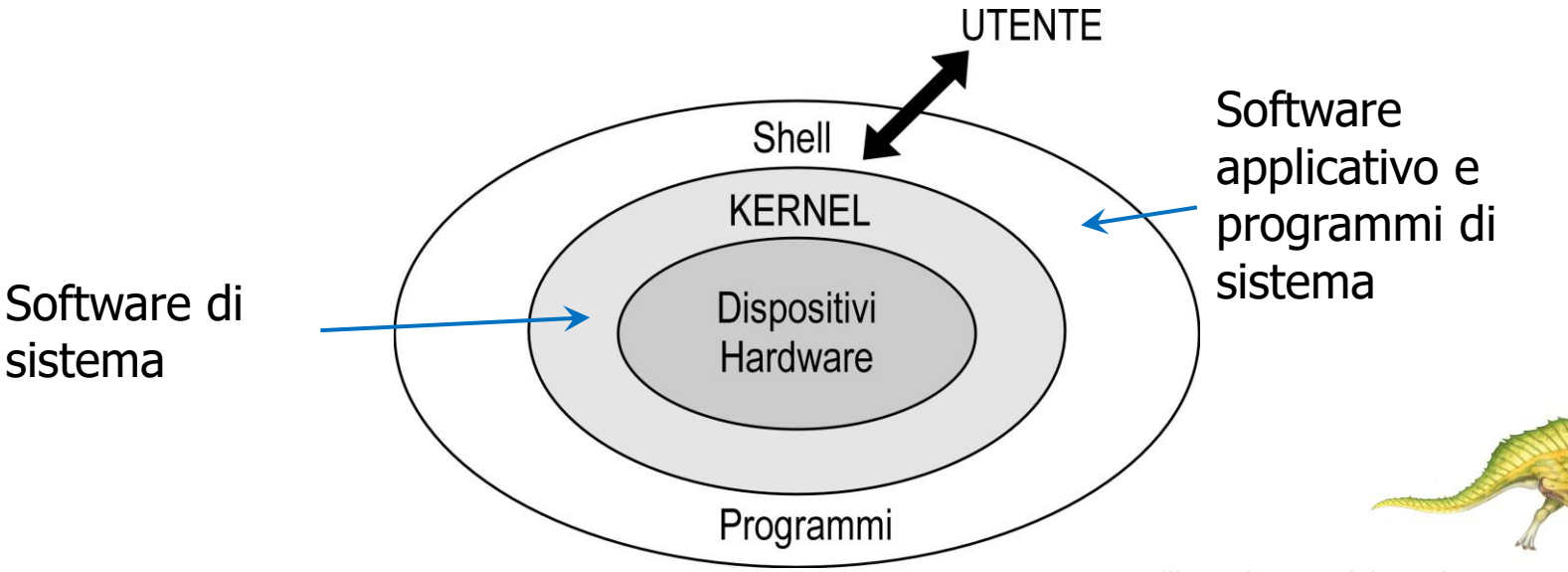
Programmi di sistema, associati al SO, ma non facenti parte del kernel

kernel



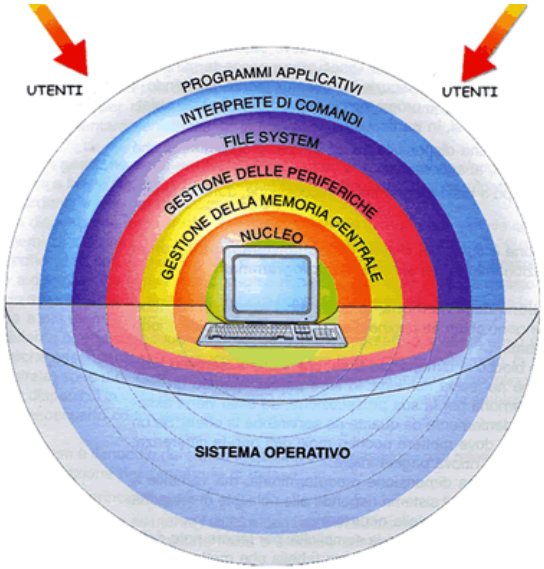
Le componenti del sistema di calcolo – 2

- La struttura del sistema di calcolo può essere schematizzata anche attraverso il "modello a cipolla", composto da una serie di *gusci concentrici*, che racchiudono l'hardware, posto al centro, e i diversi strati di software che lo gestiscono/utilizzano
- I gusci rappresentano programmi, che operano a livelli diversi di interazione uomo–macchina



Le componenti del sistema di calcolo – 3

- Anche il SO può essere descritto attraverso un modello a gusci concentrici che circondano l'hardware, a contatto diretto con il quale è il nucleo o kernel, che ingloba le funzioni più elementari eseguibili direttamente dall'hardware stesso
- All'esterno c'è invece la cosiddetta shell, che consente all'utente di accedere alle funzioni più evolute del sistema (gestione dei file, esecuzione dei programmi applicativi, operazioni complesse sulle periferiche)

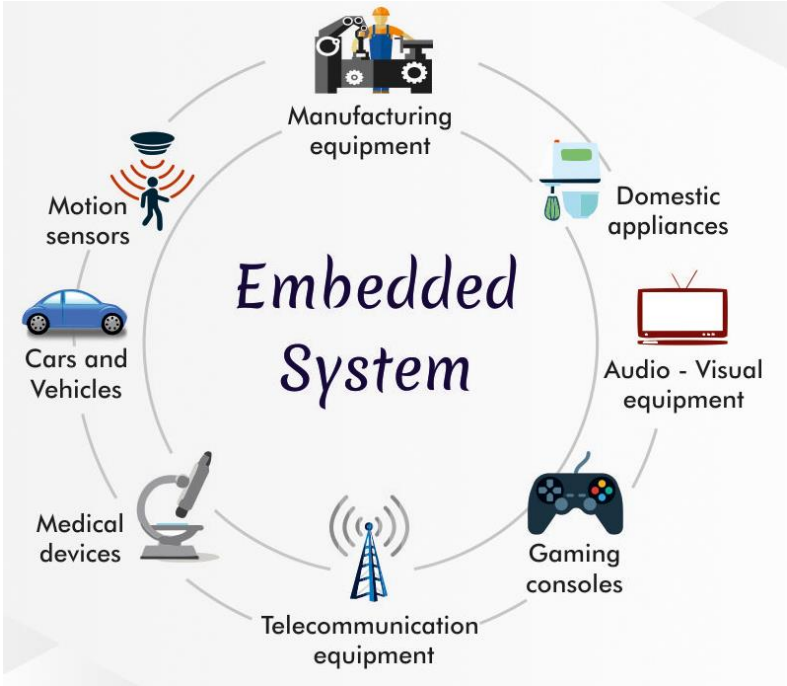


Cosa ci aspettiamo dal SO? – 1

- Dipende dai punti di vista
- Gli utenti, dal proprio computer personale, si aspettano un sistema conveniente, facile da usare e dotato di buone prestazioni
  - Non si preoccupano della gestione delle risorse
- Ma gli *shared computer*, come i mainframe o i server devono "accontentare" tutti i loro utenti
- Gli utenti dei sistemi cosiddetti "dedicati" utilizzano prevalentemente risorse locali, ma anche risorse condivise, accedendo a server
  - In questi casi una gestione delle risorse equa e regolamentata è fondamentale per il buon funzionamento del sistema di calcolo

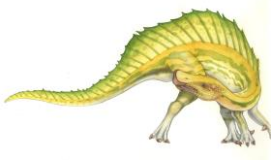
Cosa ci aspettiamo dal SO? – 2

- I device mobili, quali tablet e smartphone, non dispongono di grandi risorse, che vengono ottimizzate dal SO per fornire usabilità e basso consumo energetico
  - Interfacce utente tattili e riconoscimento vocale (Siri, Cortana, Google Assistant, Alexa)
- Certi computer, come i sistemi embedded, presenti negli elettrodomestici o nelle automobili, hanno SO non dotati di interfaccia utente
  - Elaborazione senza l'intervento dell'utente



Definizione di sistema operativo

- Il SO funge da allocatore di risorse
  - Gestisce tutte le risorse hardware/software (programmi e dati) del sistema di calcolo
  - Arbitra i conflitti per l'allocazione di risorse in base ad una politica equa ed efficiente
- Il SO è un programma di controllo
  - Controlla l'esecuzione dei programmi utente per prevenire errori e/o uso improprio delle risorse del sistema

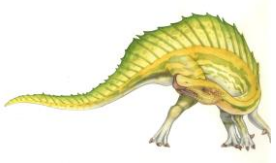


Esempio: il SO come gestore risorse – 1

- Si consideri un ristorante con un capo–cuoco (che dirige la cucina) e i suoi aiutanti, camerieri e clienti:
  - I clienti scelgono un piatto dal menù
  - Un cameriere prende l'ordine e lo consegna al capo–cuoco
  - Il capo–cuoco riceve l'ordine e assegna uno o più aiutanti alla preparazione del piatto
  - Ogni aiutante si dedicherà alla preparazione di un piatto, il che potrà richiedere più attività diverse
  - Il capo–cuoco supervisiona la preparazione dei piatti e gestisce le risorse (limitate) disponibili

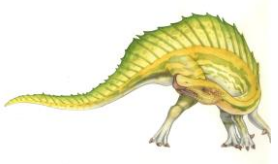
Esempio: il SO come gestore risorse – 2

- Il capo–cuoco è il sistema operativo!
  - I clienti sono gli **utenti**
  - Le ricette associate ai piatti sono i **programmi**; gli ingredienti sono i **dati**
  - Il menù ed il cameriere costituiscono l'**interfaccia** verso il SO (grafica e non)
  - Gli aiutanti sono i **processi** (eventualmente esercitanti più attività – **thread**)
  - La cucina è il **computer**; pentole, fornelli, etc., sono le componenti hardware



Esempio: il SO come gestore risorse – 3

- Problemi del capo–cuoco:
  - Esecuzione fedele delle ricette
  - Allocazione/utilizzo efficiente delle risorse esistenti (aiutanti, fornelli, ingredienti, etc.)
  - Coordinamento efficiente degli aiutanti
  - Licenziamento degli aiutanti che non si comportano secondo le regole
- Problemi del sistema operativo:
  - Esecuzione dei programmi utente
  - Efficienza nell'uso delle risorse (processori, memoria, dischi, dati, etc.)
  - Coordinamento dei processi
  - Protezione nell'uso delle risorse (e terminazione dei processi che ne abusano)



Definizione di sistema operativo (cont.)

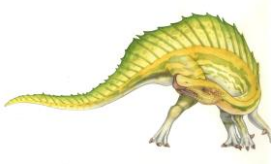
- Il termine Sistema Operativo descrive "ruoli diversi" a causa della miriade di hardware design e usi dei sistemi di calcolo
  - SO presenti negli elettrodomestici (Internet of Things), nelle auto, nelle navi, nei veicoli spaziali, nelle piattaforme per videogiochi, nei decoder, nei sistemi di controllo industriale…
  - Nati quando i computer, originariamente utilizzati solo per scopi militari e per usi governativi (decifrazione di codici, balistica, censimenti), sono diventati *general purpose* e hanno richiesto la gestione delle risorse hardware e il controllo dei programmi

Definizione di sistema operativo (cont.)

- Non esiste una definizione universalmente accettata di sistema operativo
- Tuttavia, "tutto ciò che ti fornisce il rivenditore quando gli chiedi un sistema operativo" è una buona approssimazione
  - Altamente variabile però…
- "Il sistema operativo — o meglio, il kernel del SO — è l'unico programma sempre in esecuzione quando il computer è acceso"
  - Gli altri programmi sono software di sistema o applicativi
    - Software di sistema: software ausiliario e di servizio, abitualmente presente ed utilizzato insieme al SO
    - Applicativi: strumenti di office automation, applicazioni aziendali, ambienti di sviluppo, giochi, etc.

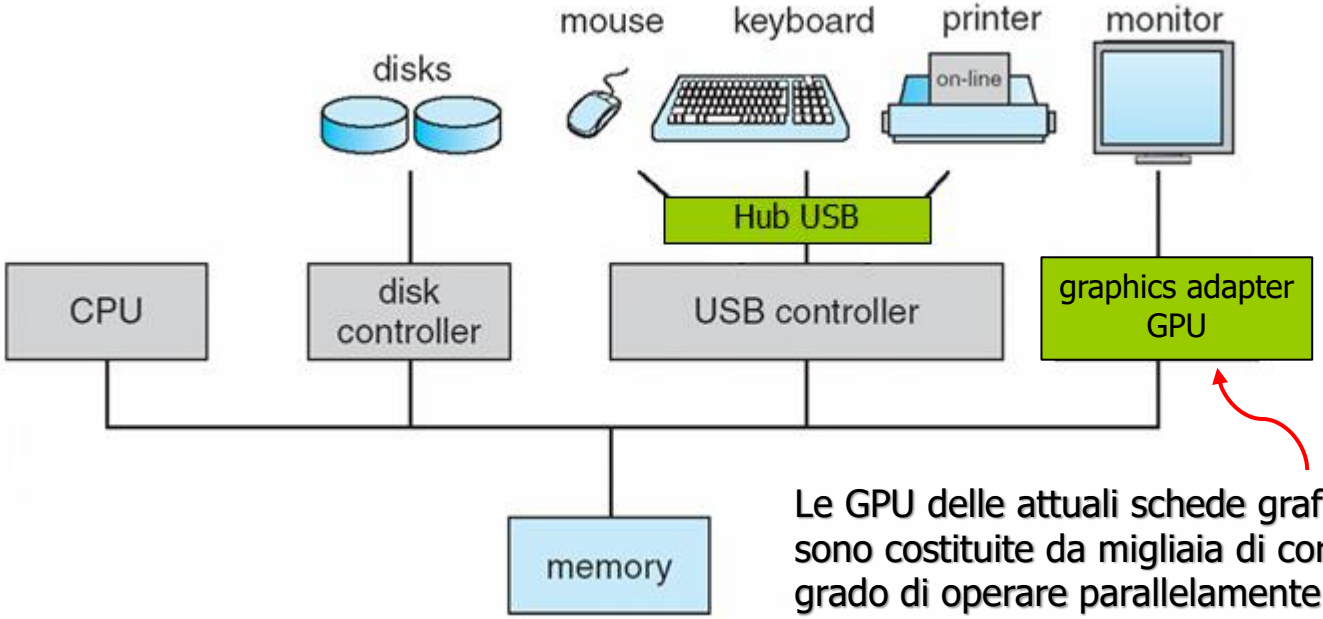
Sistemi operativi per dispositivi mobili

- I SO per dispositivi mobili non sono costituiti esclusivamente da kernel e programmi di sistema, ma anche dal **middleware**
- Il **middleware** può considerarsi una collezione di ambienti software per gli sviluppatori di applicazioni
- Sia **iOS** di Apple che **Android** dispongono di un middleware che supporta lo sviluppo di **app** multimediali, grafiche e database–oriented



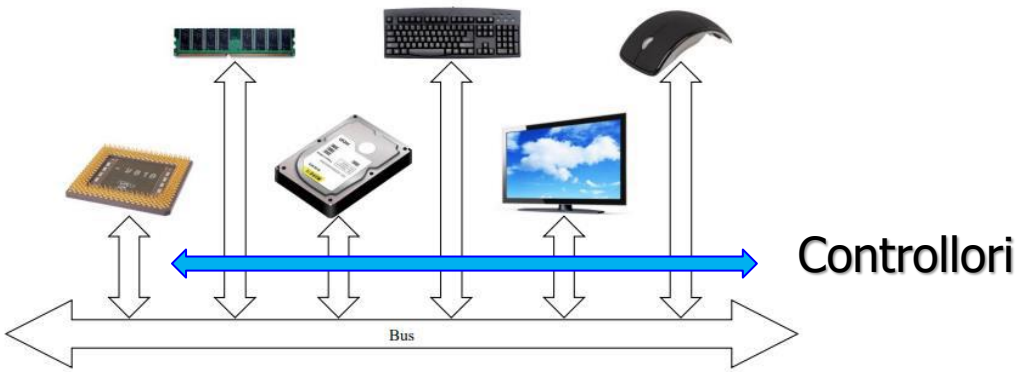
Architettura del sistema di calcolo – 1

- Una o più **CPU** e **GPU** e diversi **controllori di device** connessi ad un bus comune, che permette l'accesso ad una memoria condivisa
- Ciascun controllore si occupa di un particolare dispositivo fisico e può gestire una o più unità ad esso connesse



Architettura del sistema di calcolo – 2

- Un controllore dispone della propria memoria interna, detta *buffer*, e di un insieme di registri specializzati
- Per ogni controllore, il SO mette a disposizione un driver del dispositivo, che gestisce la specificità del controllore (e del dispositivo) e funge da interfaccia uniforme con il resto del sistema
- Esecuzione concorrente nelle CPU e nei controllori di device, che competono per garantirsi cicli di memoria
- Il controllore della memoria si occupa della sincronizzazione degli accessi

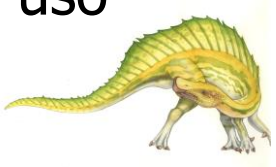


Firmware e avviamento

- Il programma di bootstrap viene caricato all'accensione del computer, o in fase di reebot, dall'UEFI (per *Unified Extensible Firmware Interface*) o, nei computer più datati, da BIOS
- Sostanzialmente, UEFI ha inglobato, come specifico componente, il BIOS, non lo ha sostituito
- Il BIOS continua a essere il firmware incorporato nella scheda madre ma l'UEFI, via interfaccia grafica, ne sovrintende l'esecuzione
  - Il BIOS è tipicamente memorizzato nella ROM o nella EEPROM e viene identificato con il termine di firmware
  - Lo UEFI può essere considerato come il successore del BIOS e costituisce comunque un'interfaccia posta tra l'hardware ed il SO

UEFI

- Infatti, rispetto al BIOS, UEFI abbandona la scarna interfaccia testuale per abbracciare una più accattivante interfaccia grafica che supporta direttamente l'utilizzo del mouse
- Mentre il BIOS permetteva la gestione di comandi a 16 bit con la possibilità di indirizzare solo 1 MB di memoria, UEFI può operare in modalità a 32 o 64 bit con la possibilità di indirizzare un maggior quantitativo di memoria RAM in modo tale da assolvere a compiti nettamente più complessi
- Fornisce, infatti, strumenti per la diagnostica e il ripristino dei dati, servizi di crittografia e funzionalità per la gestione dei consumi
- UEFI può essere anche del tutto indipendente dall'architettura hardware e può integrare driver per componenti anch'essi indipendenti dalla tipologia di processore in uso

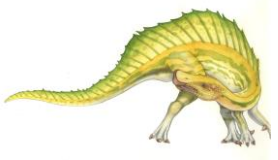


La fase di avviamento – 1

- Il programma di bootstrap ha lo scopo principale di caricare il *bootloader*, che è memorizzato su memoria di massa
  - Il bootloader controlla tutte le componenti del sistema — questa procedura viene chiamata **POST** (Power–On Self–Test) — e le inizializza
  - Procede al caricamento del kernel del SO e ne lancia l'esecuzione
- Quando si accende un elaboratore, occorre attendere alcuni istanti per poter iniziare a lavorare: durante questa pausa il computer carica il kernel e i programmi di sistema, che garantiscono i servizi fondamentali per l'esecuzione dei programmi utente

La fase di avviamento – 2

- I programmi di sistema, detti anche *daemon*, vengono eseguiti in background e restano in esecuzione per tutto il tempo in cui è in esecuzione il kernel
- Rimangono in ascolto di eventi specifici e forniscono, in risposta, dei servizi agli utenti
- In Linux, il primo processo di sistema è **`systemd`**, che funge da gestore di tutti gli altri demoni

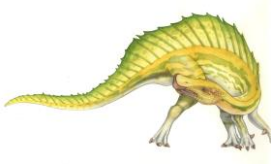


Operatività del sistema di calcolo

- La CPU e i device di I/O possono eseguire operazioni concorrenti
- Ciascun controllore gestisce un particolare tipo di periferica
- Ciascun controllore gestisce un buffer locale ed è responsabile del trasferimento dei dati dalle periferiche ad esso connesse al buffer
- Ogni controllore ha un driver di dispositivo — facente parte del sistema operativo — ad esso dedicato
- La CPU sposta i dati dalla memoria principale verso i buffer locali e viceversa
- L'I/O avviene effettivamente utilizzando i buffer dei controllori come memorie di transito
- Il controllore informa la CPU quando ha portato a termine l'operazione di I/O, causando un *interrupt*

Gli interrupt

- Al verificarsi di un interrupt, il controllo viene trasferito all'opportuna routine di gestione utilizzando il vettore degli interrupt (residente nella memoria bassa), che contiene gli indirizzi di tutte le routine di servizio
  - Si salva inoltre l'indirizzo dell'istruzione "interrotta" – e lo "stato" della CPU – dalla quale dovrà riprendere l'esecuzione al termine della gestione dell'interruzione
- Durante la gestione di un interrupt, altri eventuali interrupt giunti al sistema possono essere momentaneamente "disabilitati"
- Una *trap* (o eccezione) è un interrupt generato via software, causato da un errore o da una richiesta utente (effettuata tramite una *system call*)
- Il SO è *interrupt driven*



Interrupt timeline

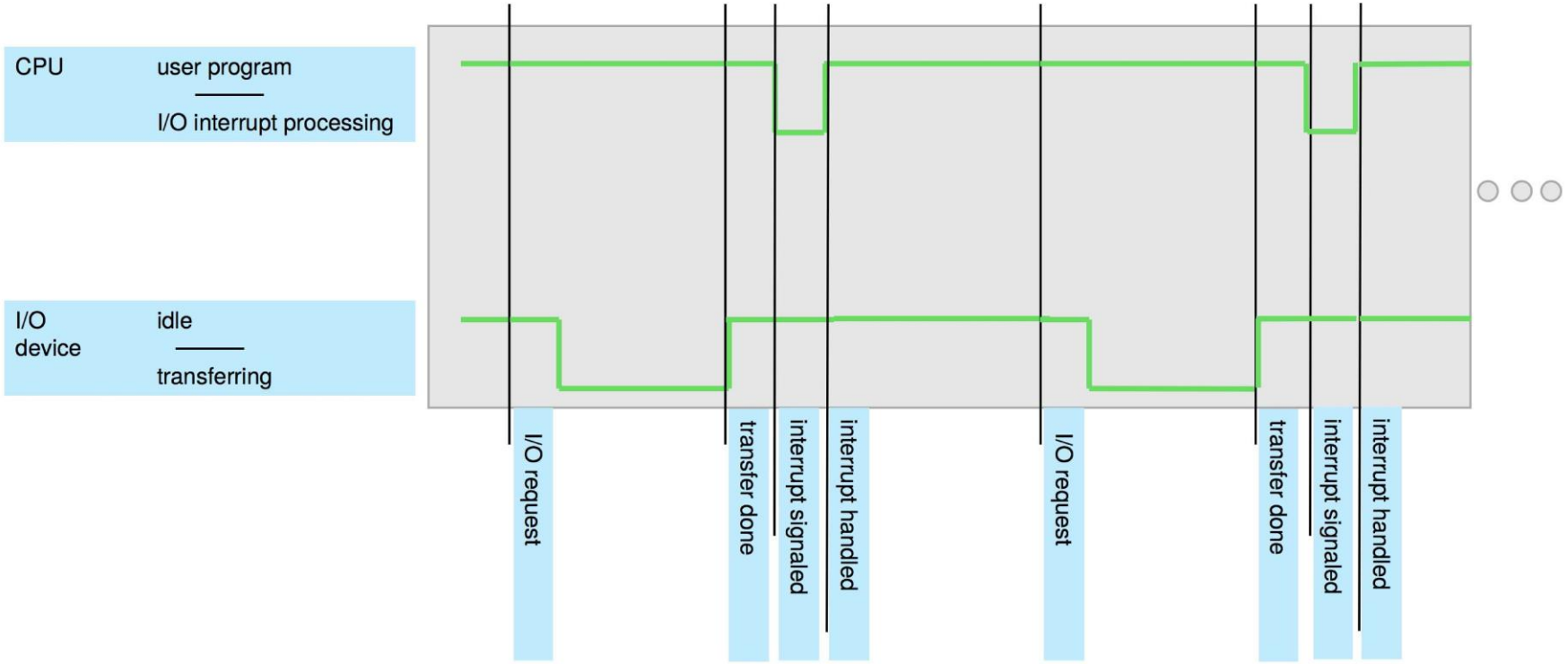
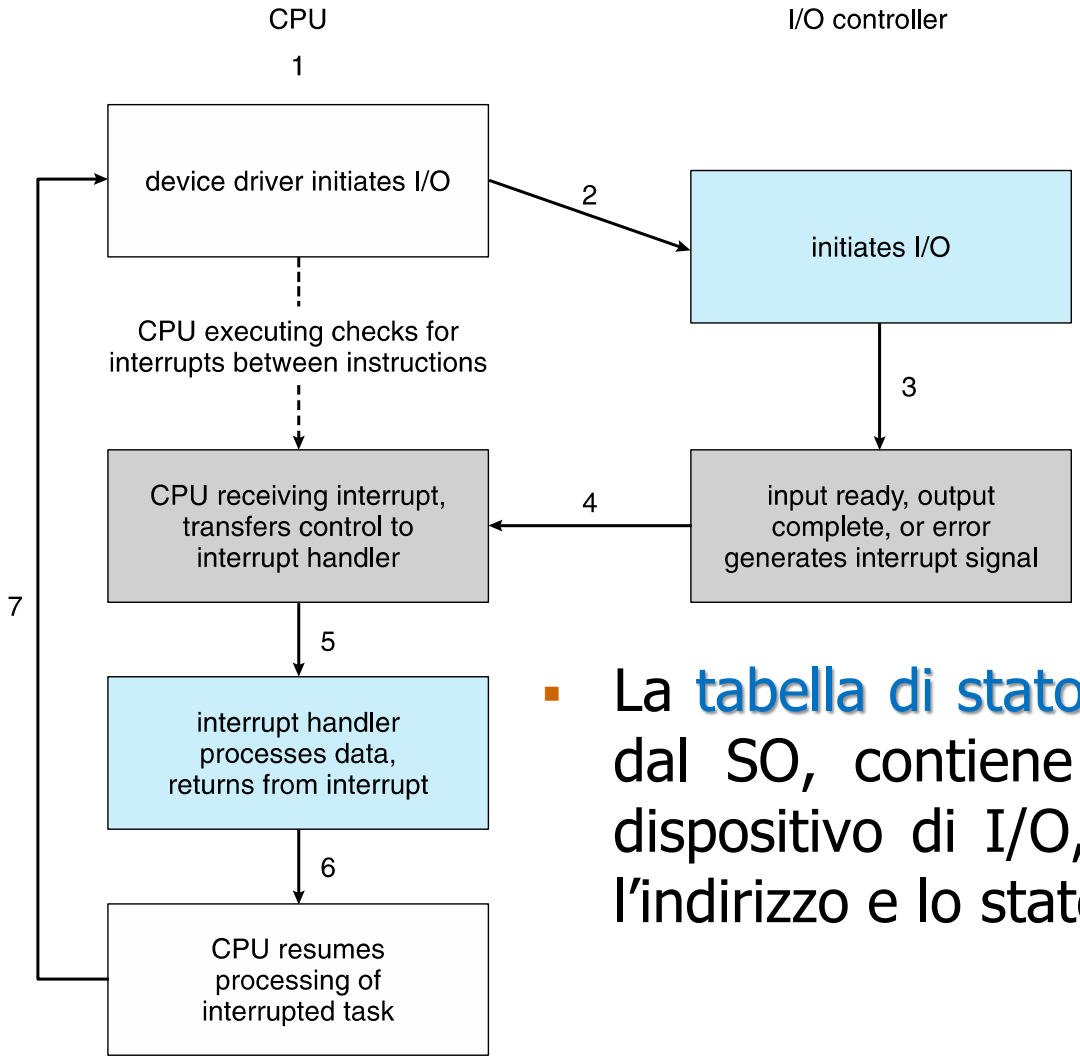


Diagramma temporale delle interruzioni per un programma che invia dati in output

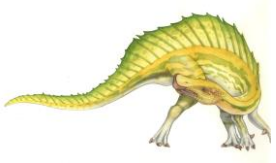
Gli interrupt: implementazione

- L'hardware della CPU dispone di una o più linee (dedicate) di richiesta di interruzione
- Il gestore delle interruzioni ne rileva la natura, salva le informazioni di stato, invoca la routine di servizio opportuna e, infine, esegue un'istruzione di ritorno e ripristina lo stato della CPU
- Nei sistemi di ultima generazione, hardware dedicato al controllo delle interruzioni
  - Efficienza nel recuperare la routine di servizio
  - Possibilità di posticipo della gestione degli interrupt durante un'elaborazione critica
  - Interruzioni mascherabili, cioè temporaneamente disattivabili (inviate da controllori), e non mascherabili (condizioni di errore) → due *interrupt request line*
  - Interruzioni multilivello, con diverse priorità, per reagire con l'urgenza appropriata

Ciclo di I/O interrupt driven



- La tabella di stato dei dispositivi, gestita dal SO, contiene un item per ciascun dispositivo di I/O, che ne indica il tipo, l'indirizzo e lo stato



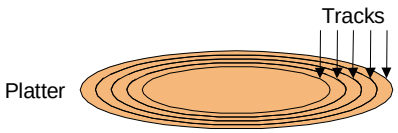
Interrupt nei processori Intel

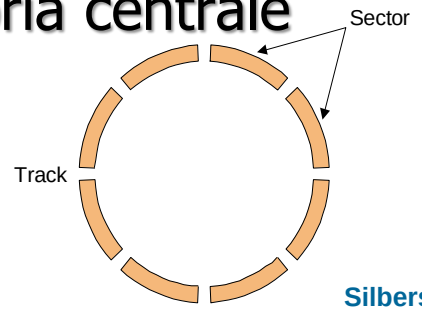
numero di vettore	descrizione
0	errore di divisione
1	eccezione di debug
2	interruzione null
3	breakpoint
4	eccezione di overflow
5	eccezione di range exceeded
6	codice operativo non valido
7	dispositivo non disponibile
8	doppio errore
9	overrun del segmento coprocessore (riservato)
10	task state segment (tss) non valido
11	segmento non presente
12	errore di stack
13	protezione generale
14	errore di pagina
15	(riservato Intel, non utilizzare)
16	errore in virgola mobile
17	controllo dell'allineamento
18	controllo della macchina
19–31	(riservato Intel, non utilizzare)
32–255	interruzioni mascherabili

Gli eventi da a 0 a 31 non sono mascherabili e servono per segnalare condizioni di errore; gli eventi da 32 a 255 vengono utilizzati per le interruzioni provenienti da dispositivi

Struttura della memoria

- **Memoria principale:** l'unico dispositivo di memoria cui la CPU può accedere direttamente (volatile, accesso random)
- **Memorie di massa:** "estensioni" della memoria principale; forniscono notevole capacità di memorizzazione non volatile
  - ⇨ **"Dischi" a stato solido** – *Non volatile memory* (NVM), memorie elettroniche non volatili, prive di organi meccanici
    - Più veloci dei vecchi hard disk
    - Estremamente diffuse, diverse tecnologie
  - ⇨ **Dischi magnetici** – "piatti" rigidi di metallo o in fibra di vetro, ricoperti di materiale magnetico
    - La superficie del disco è logicamente suddivisa in **tracce**, a loro volta suddivise in **settori**
- In entrambi i casi: il **controllore del disco** determina l'interazione logica fra device e memoria centrale

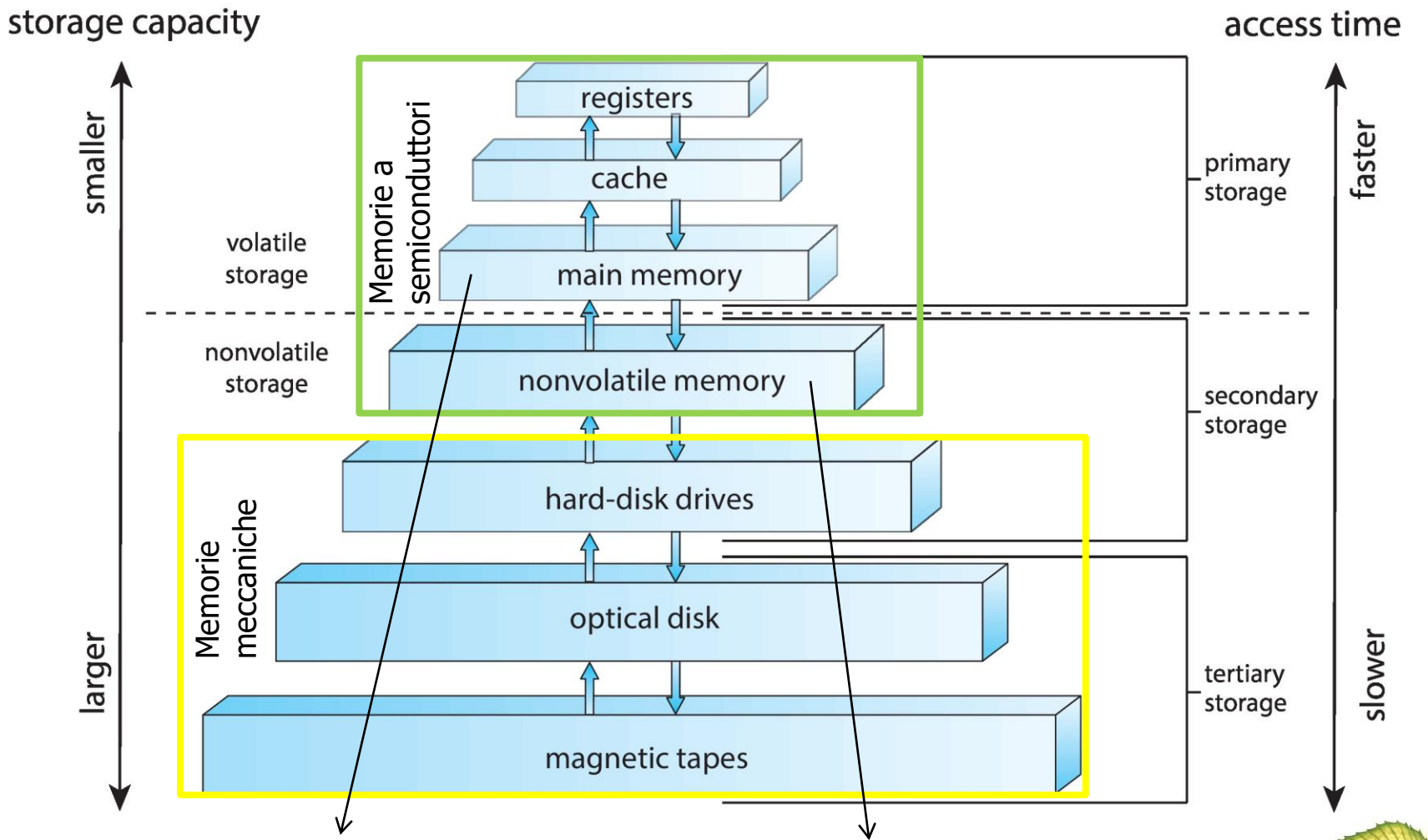




Gerarchia dei dispositivi di memoria – 1

- I dispositivi di memoria sono organizzati gerarchicamente in base a:
  - Velocità di accesso
  - Costo (al byte)
  - Volatilità
- *Caching* – produzione di una copia dell'informazione in un dispositivo di memoria più veloce (situato più in alto nella gerarchia delle memorie); la memoria principale costituisce una cache per le memorie secondarie

Gerarchia dei dispositivi di memoria – 2



Memoria DRAM (Dynamic Random Access Memory)

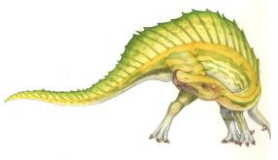
Memoria flash (NAND): Più lenta della DRAM; non necessita di alimentazione esterna

Caching – 1

- Principio informatore fondamentale dei moderni sistemi di calcolo, attuato a vari livelli (hardware, SO, software)
- L'informazione in uso viene copiata temporaneamente da un dispositivo di memoria più lento ad uno più veloce
- Il dispositivo più veloce, che funge da cache, viene controllato per verificare se l'informazione d'interesse è già presente, nel qual caso…
  - …l'informazione viene reperita direttamente dalla cache
  - Altrimenti, i dati vengono preventivamente copiati nella cache e quindi acceduti

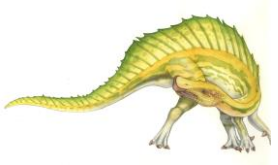
Caching – 2

- La cache ha dimensioni inferiori rispetto al supporto di memoria dal quale si effettua il caching
  - La gestione della cache, data la capacità limitata dei dispositivi di caching, rappresenta un problema di progettazione fondamentale
  - Un'appropriata selezione delle dimensioni e dei criteri di aggiornamento della cache può garantire che una percentuale variabile dall'80% al 99% degli accessi si limiti alla cache, con un notevole incremento delle prestazioni del sistema



Struttura dell'I/O – 1

- Una percentuale cospicua del codice del SO è dedicata alla gestione dell'I/O, perché:
  - l'interazione con i dispositivi di I/O è fondamentale per il progetto di un sistema affidabile ed efficiente
  - vi è una grande variabilità nei dispositivi di ingresso/uscita
- Ciascun controllore si occupa di un particolare tipo di dispositivo (può gestire una o più unità ad esso connesse)
- Il driver del dispositivo è il "sistema operativo" del controllore

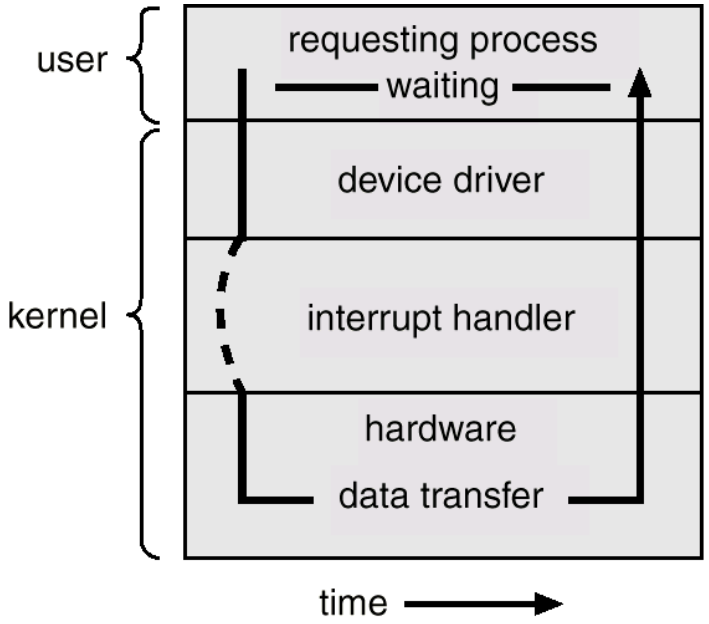


Struttura dell'I/O – 2

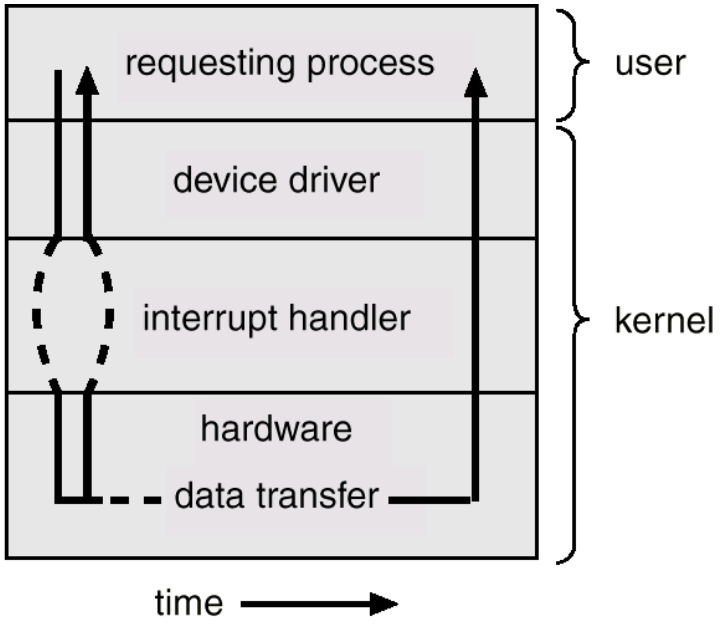
- In seguito ad una richiesta di I/O, da parte di un processo utente, si verifica un'interruzione
- Dopo l'inizio dell'operazione di I/O, il flusso di esecuzione può seguire due percorsi distinti
  - Nel caso più semplice, si restituisce il controllo al processo utente solo dopo il completamento dell'operazione di I/O (*I/O sincrono*)
    - La CPU è mantenuta in stato idle da una *wait*
    - Ad ogni istante, si può avere una sola richiesta di I/O in esecuzione
  - Altrimenti, si può prevedere la restituzione immediata del controllo al (ad altro) processo utente: in questo modo l'I/O può proseguire mentre il sistema esegue altre operazioni (*I/O asincrono*)

I/O sincrono e asincrono

Sincrono

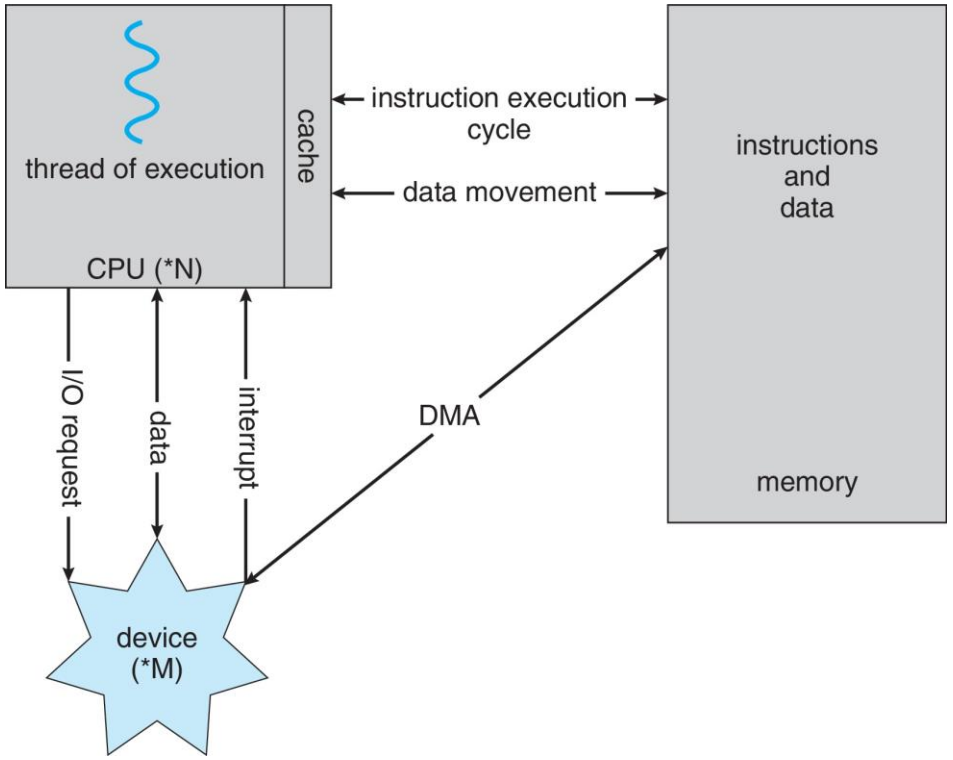


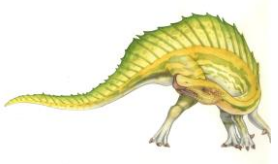
Asincrono



Direct Memory Access (DMA)

- Utilizzato per dispositivi di I/O ad alte prestazioni, in grado di trasmettere informazioni a velocità prossime all'accesso alla memoria
- Il controller del dispositivo trasferisce blocchi di dati dal buffer locale direttamente nella memoria principale, senza l'intervento della CPU
- Viene generato un solo interrupt per blocco, anziché un interrupt per byte



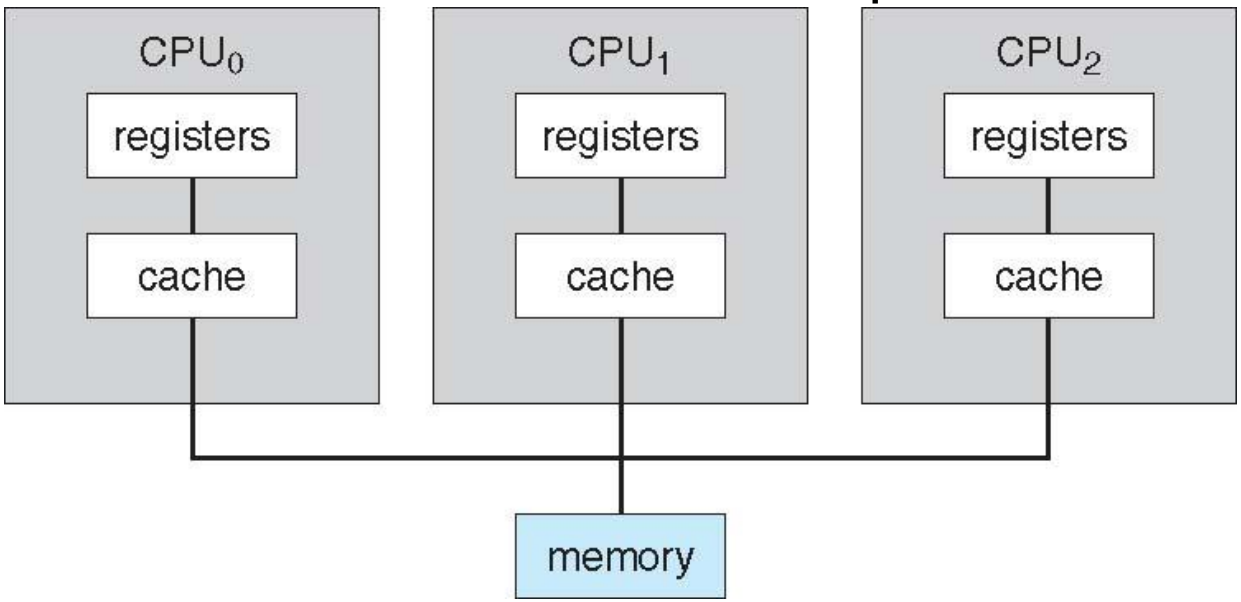


Architettura degli elaboratori – 1

- Fino a due decenni fa, la maggior parte dei sistemi di calcolo utilizzava una singola CPU general–purpose
  - …pur inglobando molti microprocessori special–purpose
- A partire dal 2005, i sistemi multiprocessore, detti anche sistemi paralleli o multicore, hanno iniziato a dominare il mondo della computazione
  - Maggiore throughput (maggior numero di processi terminati nell'unità di tempo)
  - Migliore economia di scala (prezzi più bassi rispetto a più sistemi monoprocessore — condivisione di periferiche, memorie di massa, sistemi di alimentazione)
  - Incremento dell'affidabilità (i guasti non bloccano il sistema, *graceful degradation* o *fault tolerance*)

Architettura degli elaboratori – 2

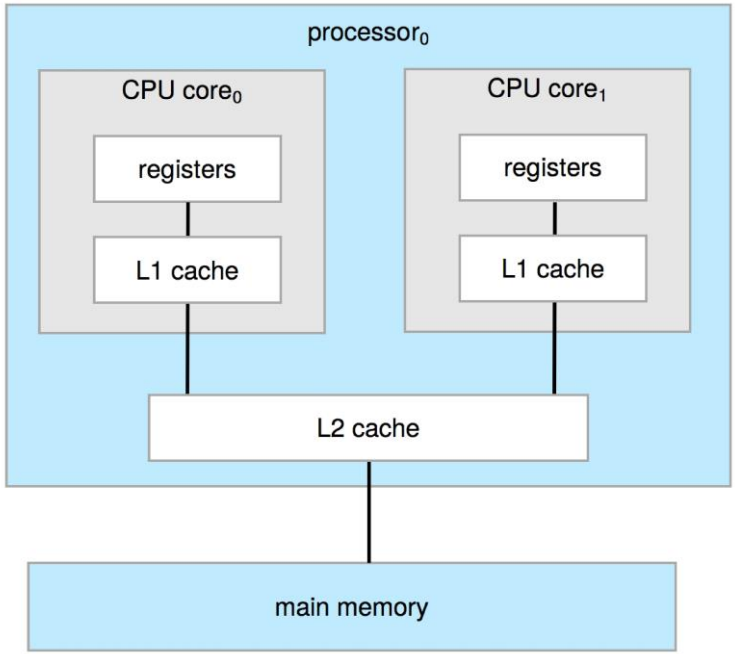
- I sistemi multiprocessore attualmente in uso permettono…
  - la multielaborazione asimmetrica — ad ogni processore viene assegnato un compito specifico; un processore principale sovrintende all'intero sistema
  - la multielaborazione simmetrica — ogni processore è abilitato all'esecuzione di tutte le operazioni del sistema



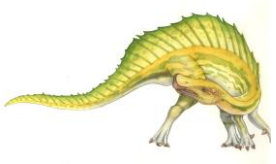
Architettura per il multiprocessing simmetrico (SMP)

Sistemi multicore

- Nei sistemi multicore si raggruppano diverse unità di calcolo — *core* — in un singolo chip
  - Più efficienti, perché le comunicazioni, sul singolo chip, sono più veloci
  - Un chip multicore usa molta meno potenza di diversi chip single core

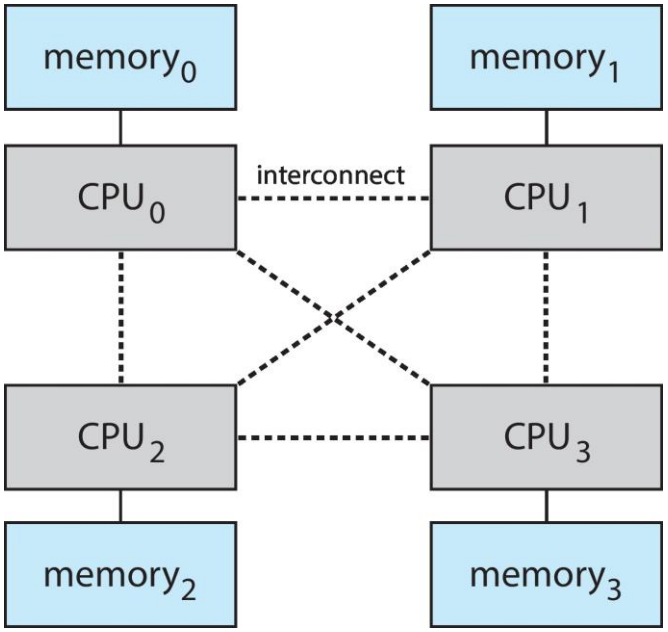


Architettura dual–core



Non–Uniform Memory Access System (NUMA)

- Architettura NUMA: ogni CPU è dotata di una propria memoria locale; le CPU sono collegate da un'interconnessione di sistema condivisa
  - Elimina il collo di bottiglia dovuto al bus di sistema comune
  - Migliore economia di scala
  - Accesso non un uniforme alle memorie



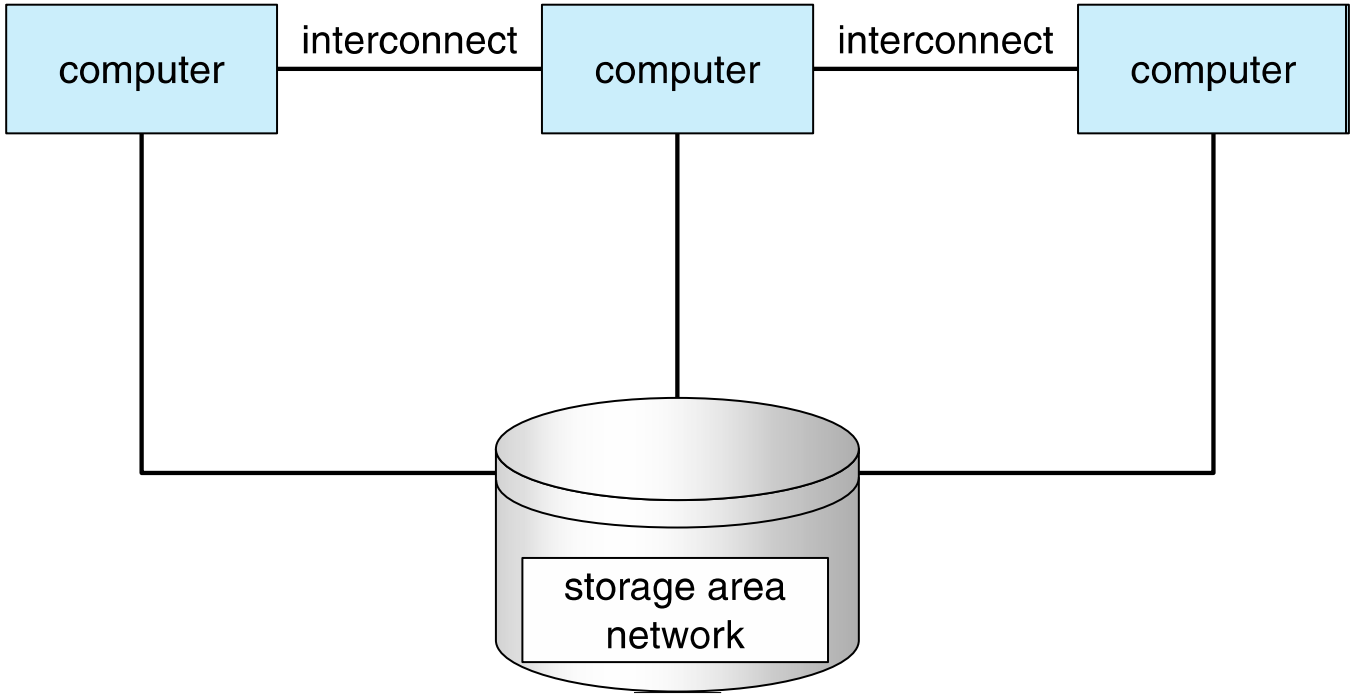
- Il SO può ottimizzare le prestazioni con un attento scheduling della CPU ed un'accurata gestione della memoria

Cluster di elaboratori – 1

- Simili ai sistemi multiprocessore, ma composti da due o più calcolatori completi, detti *nodi*, collegati tra loro
  - I cluster sono sistemi debolmente accoppiati
  - Normalmente, memoria secondaria condivisa tramite SAN (storage–area network)
- Notevole fault tolerance
  - Clustering asimmetrico: un calcolatore rimane nello stato di attesa attiva (in *hot–standby mode*), mentre gli altri eseguono le applicazioni
    - ➡ Il calcolatore in stand–by controlla gli altri nodi e si attiva nel caso di malfunzionamenti
  - Clustering simmetrico: tutti i nodi eseguono le applicazioni e si controllano reciprocamente

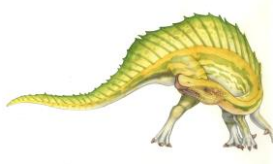
Cluster di elaboratori – 2

- I cluster vengono impiegati nell'high–performance computing (HPC)
  - Le applicazioni devono essere parallelizzate, cioè scritte per trarre vantaggio dall'architettura



Struttura del sistema operativo – 1

- La multiprogrammazione è fondamentale per l'efficienza del sistema
  - Un singolo processo non può tenere occupati contemporaneamente la CPU ed i device di I/O
  - Con la multiprogrammazione, più processi (codice e dati) sono memorizzati contemporaneamente nella memoria principale, così da mantenere la CPU sempre occupata
  - Il processo che verrà eseguito, ed otterrà quindi l'utilizzo esclusivo della CPU, viene selezionato mediante un algoritmo di *CPU scheduling*
  - Quando un processo è in attesa di qualche evento (per esempio, un servizio di I/O), il SO seleziona un nuovo processo da eseguire

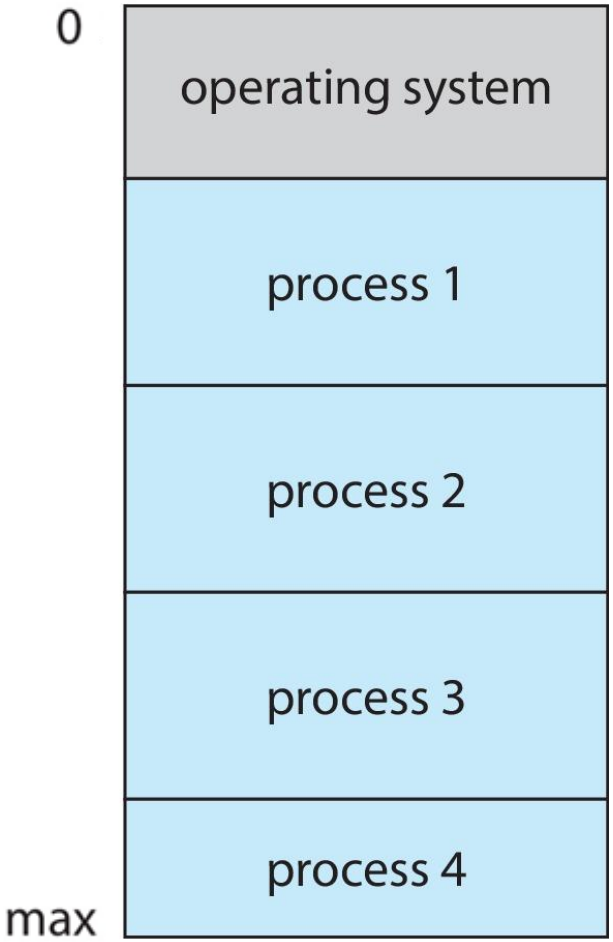


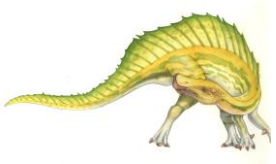
Struttura del sistema operativo – 2

- Il *timesharing* (o *multitasking*) è l'estensione logica della multiprogrammazione, nella quale la CPU passa così rapidamente dall'esecuzione di un processo a quella del successivo che gli utenti possono interagire con i loro processi durante la fase di running —*interactive computing*
  - Il tempo di risposta deve essere << 1 secondo
  - Ciascun utente ha almeno un programma allocato in memoria centrale "in esecuzione" ⇨ un processo
  - Se più processi sono pronti per essere eseguiti, deve essere implementato un algoritmo per lo scheduling della CPU
  - Se la memoria non è sufficiente a contenere tutti i processi, mediante *swapping*, essi vengono spostati fuori/dentro la memoria principale in base al loro stato di esecuzione effettivo
  - La metafora della memoria virtuale garantisce l'esecuzione di processi non contenuti completamente nella memoria principale e più grandi della memoria fisica presente nel sistema

Multiprogrammazione

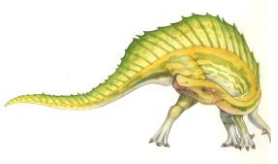
Configurazione della memoria per un sistema multiprogrammato





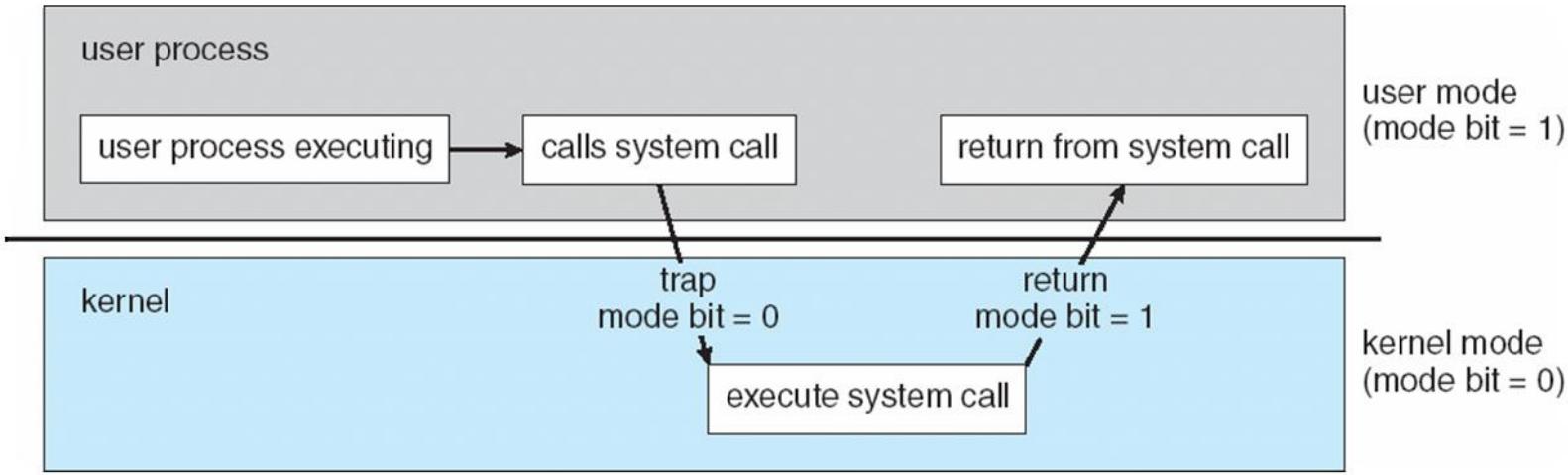
Attività del sistema operativo – 1

- Il Sistema Operativo è interrupt–driven
  - Interrupt hardware provenienti dai device
  - Errori nel software o richieste di servizi creano situazioni di eccezione o trap
    - Divisioni per zero, processi in loop (riempimento dello stack), processi che tentano di modificare lo spazio di memoria dedicato ad altri processi o al sistema operativo
    - Richieste di servizio al SO
- La modalità operativa *dual–mode* permette al SO di proteggersi e di proteggere le varie componenti del sistema di calcolo



Attività del sistema operativo – 2

- *User mode* e *kernel mode*
  - *Mode bit* cablato in hardware
    - Il valore assunto dal mode bit distingue le due situazioni in cui il sistema esegue codice utente o codice kernel; le istruzioni privilegiate possono essere eseguite soltanto in modalità kernel
    - Ogni *system call* effettua il passaggio in modalità kernel; il ritorno dalla chiamata riporta il sistema in modalità utente



Attività del sistema operativo – 3

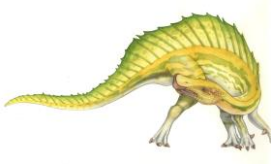
- Il concetto di modalità può essere esteso oltre il dual mode (utilizzo di più bit di modo)
  - Le CPU hanno spesso una modalità distinta per gestire la virtualizzazione, ovvero per indicare che il gestore della macchina virtuale ha il controllo del sistema (più privilegi dei processi utente, ma meno del kernel del SO ospitante)
  - Possibilità di modalità distinte per le diverse componenti del kernel
  - Esempio: le CPU Intel 64 supportano quattro livelli di privilegio

Passaggio da modo utente a modo kernel

- Per prevenire processi che eseguono cicli infiniti senza più restituire il controllo al SO ⇨ Timer
  - Si realizza mediante un clock e un contatore
  - Emissione di un interrupt dopo un dato periodo di tempo
- Il SO inizializza il contatore al tempo massimo (stimato) di esecuzione del processo
  - Il contatore viene decrementato ad ogni impulso del clock
  - Quando il contatore assume valore zero, si genera un interrupt, che restituisce il controllo al SO
  - Il timer viene impostato ogni volta che il processo ottiene l'uso della CPU, in base al tempo di esecuzione assegnatogli, e ne garantirà l'interruzione, con restituzione del controllo al SO, in caso di superamento

Gestione dei processi – 1

- Un *processo* è un programma in esecuzione
  - Il programma è un'*entità passiva*, il processo un'*entità attiva*
- Il processo è l'unità di lavoro del sistema di calcolo
- Un processo necessita di alcune risorse per assolvere al proprio compito: tempo di CPU, memoria, file, dispositivi di I/O e dati di inizializzazione
- La terminazione di un processo prevede il recupero di tutte le risorse riutilizzabili ad esso precedentemente allocate

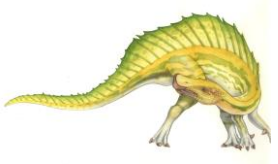


Gestione dei processi – 2

- I processi *single–thread* hanno un unico *program counter*, che indica la locazione in memoria della prossima istruzione da eseguire
  - Le istruzioni di ogni processo vengono eseguite sequenzialmente, una dopo l'altra, fino al termine del processo
- I processi *multi–thread* hanno un program counter per ogni thread
- Normalmente, in un sistema, vi sono molti processi di uno o più utenti, e alcuni processi del SO, che vengono eseguiti in concorrenza su una o più CPU
  - La concorrenza è ottenuta effettuando il *multiplexing* delle CPU fra i vari processi/thread

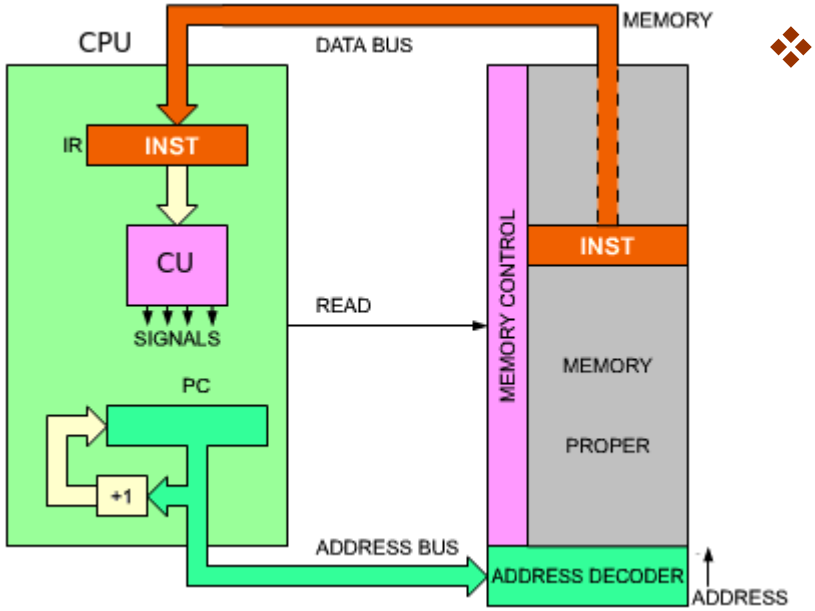
Gestione dei processi – 3

- Il SO è responsabile delle seguenti attività relative alla gestione dei processi:
  - Creazione e cancellazione di processi (utente e di sistema)
  - Scheduling dei processi/thread sulla/e CPU
  - Sospensione e riattivazione di processi
  - Fornire meccanismi per
    - la comunicazione fra processi
    - la sincronizzazione dei processi
    - la gestione del *deadlock*



Gestione della memoria centrale – 1

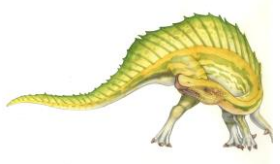
- La memoria centrale è un vettore di miliardi di parole, ciascuna con un proprio indirizzo
- È un deposito di dati rapidamente accessibili, condiviso dalla CPU e dai dispositivi di I/O
  - …ma è lenta rispetto alla CPU
- La CPU legge le istruzioni dalla memoria durante la fase di *fetch*, oltre a leggere e scrivere i dati



- Il modulo per la gestione della memoria determina cosa è in essa contenuto ad un certo istante, per ottimizzare l'utilizzo della CPU ed il tempo di risposta del sistema agli utenti

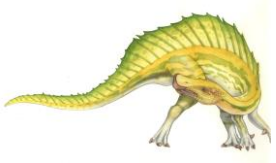
Gestione della memoria centrale – 2

- Per eseguire un programma, tutte le sue istruzioni (o, almeno, una parte di esse) devono risiedere in memoria centrale; ugualmente dicasi per i dati
- Il SO è responsabile delle seguenti attività connesse alla gestione della memoria centrale:
  - Tenere traccia di quali parti della memoria sono attualmente usate e da chi
  - Decidere quali (parti di) processi caricare in memoria quando vi è spazio disponibile
  - Allocare e deallocare lo spazio di memoria secondo necessità



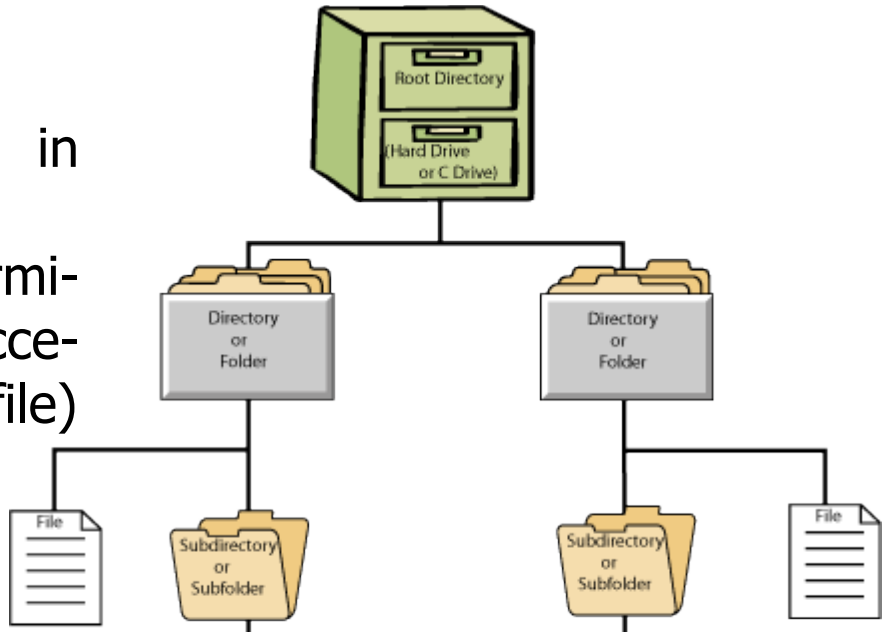
Gestione della memoria secondaria

- Il SO garantisce una visione logica uniforme del processo di memorizzazione secondaria
  - Astrae dalle caratteristiche fisiche dei dispositivi per definire un'unità di memorizzazione logica — il *file*
    - Il concetto di file è indipendente dal mezzo sul quale viene memorizzato (che ha caratteristiche proprie e propria organizzazione fisica)
    - I file possono contenere programmi e dati
    - I file di dati possono essere numerici, alfabetici, alfanumerici o binari e possono avere una struttura libera (es. file di testo) o rigidamente formattata (es. file mp3)
  - Ciascuna periferica viene controllata dal relativo *device driver*, che nasconde all'utente (e al resto del sistema) le caratteristiche fisiche variabili dell'hardware
    - Modalità e velocità di accesso, capacità, velocità di trasferimento, etc.

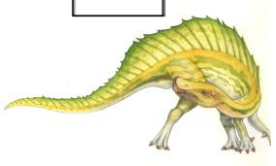


Gestione del file system – 1

- Un file è una collezione di informazioni correlate, definita dal suo creatore
- Gestione del file system
  - I file vengono organizzati in directory
  - Controlli di accesso per determinare quali utenti possono accedere e come a quali risorse (file)

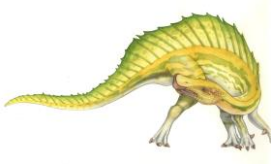


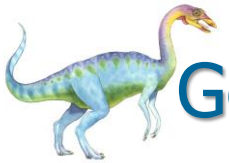

```
root@tecadmin:~/test#
root@tecadmin:~/test# ls
logs  myfile.txt  test.log
root@tecadmin:~/test#
root@tecadmin:~/test# ls -l
total 232
drwxr-xr-x 2 root root   4096 Sep  5 17:05 logs
-rw-r--r-- 1 root root     12 Sep  5 17:04 myfile.txt
-rw-r----- 1 root root 227829 Sep  5 17:04 test.log
root@tecadmin:~/test#
```



Gestione del file system – 2

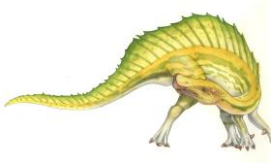
- Il SO è responsabile delle seguenti attività per la gestione dei file:
  - Creazione e cancellazione di file e directory
  - Supporto alle funzioni elementari per la manipolazione di file e directory
  - Associazione dei file ai dispositivi di memoria secondaria
  - Backup di file su dispositivi stabili di memorizzazione (memoria terziaria)





Gestione dei supporti di memoria secondaria

- La memoria centrale è volatile e troppo piccola per contenere permanentemente tutti i dati
  - Il sistema di elaborazione deve consentire l'archiviazione secondaria, per salvare i contenuti della memoria centrale
  - Gli SSD (e i dischi magnetici) costituiscono il principale mezzo di memorizzazione permanente
- La velocità di accesso alla memoria secondaria è fondamentale per le prestazioni del sistema, che risentono fortemente della bontà degli algoritmi utilizzati per il reperimento dei dati in essa contenuti
- Il SO è responsabile delle seguenti attività:
  - Creazione delle partizioni
  - Operazioni di montaggio e smontaggio
  - Gestione dello spazio libero
  - Allocazione dello spazio
  - Accesso/protezione al/del dispositivo di memoria



Gestione della memoria terziaria

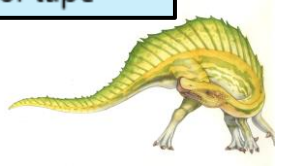
- I dispositivi di memoria terziaria (utilizzati tipicamente off–line) non devono essere particolarmente veloci
  - Le memorie terziarie sono normalmente realizzate su supporti rimovibili (ad esempio: dispositivi USB basati su memorie flash) → un unico drive utilizzato per diversi media
  - Devono comunque essere gestite in modo efficiente (dal SO o da programmi di sistema)
  - Supporto del SO: installazione/rimozione dei media (montaggio/smontaggio dei relativi file system), allocazione dei dispositivi ai processi che ne richiedono l'uso (esclusivo)

Performance dei diversi livelli di memoria

- Lo "spostamento" di dati nella gerarchia di memorie può essere esplicito (mediato dal SO) o implicito (gestito dall'hardware)
  - da disco a RAM
  - da cache a registri

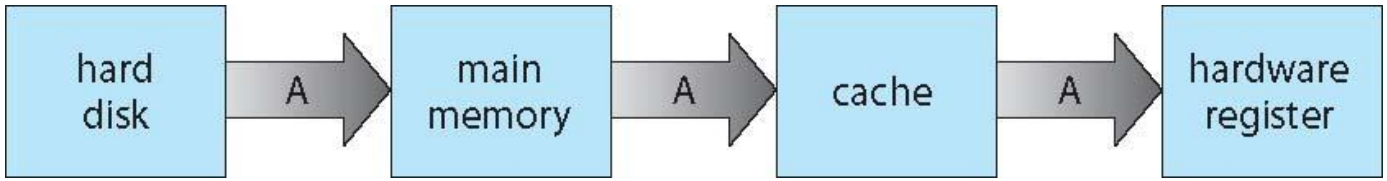
Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Dati 2019



Migrazione di un dato "A" da disco a registro

- Si voglia modificare il dato "A", residente nel file "B"
- In ambiente multitasking, è necessario assicurare che qualsiasi processo che desideri accedere ad "A", otten-ga dal sistema il valore aggiornato



  - In ambiente multiprocessore, la *coerenza della cache* si ottie-ne garantendo che l'aggiornamento di "A" in una qualsiasi cache si rifletta immediatamente in tutte le cache in cui "A" risiede (il problema si risolve a livello hardware, senza inter-vento del SO)
  - In ambiente distribuito, più copie del dato possono coesistere in memorie non volatili distinte con cache distinte, ma la coerenza deve essere comunque garantita

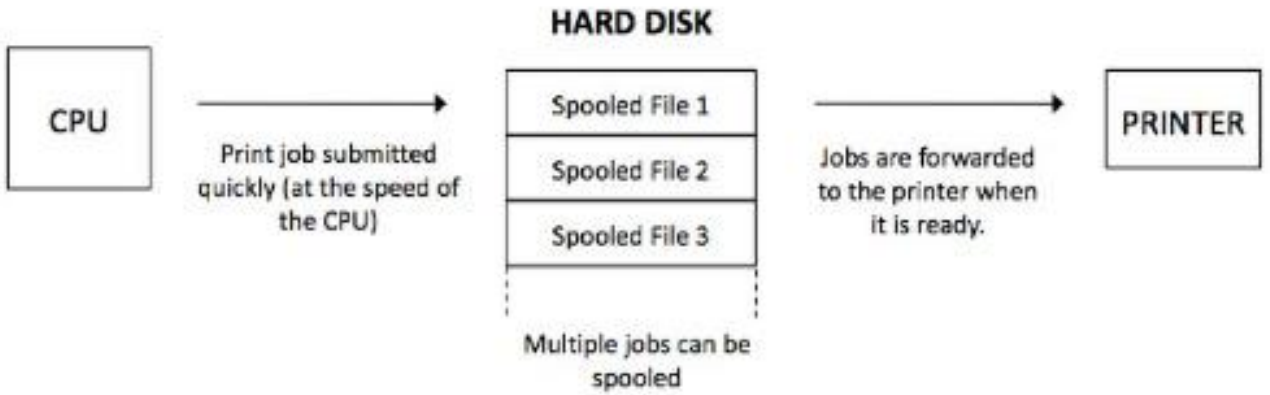
Il sottosistema di I/O – 1

- Uno degli scopi del SO consiste nel nascondere all'utente le caratteristiche dei dispositivi hardware
- Il sottosistema di I/O è formato da:
  - Una componente di gestione della memoria che include il **buffering** (la memorizzazione temporanea di dati in memorie locali alle periferiche, durante il trasferimento); il **caching** (la memorizzazione parziale dei dati in memorie ad accesso rapido – mappatura dei buffer in memoria centrale); lo **spooling** (*simultaneous peripheral operations on-line,* per la gestione dell'I/O asincrono)
  - Un'interfaccia generale per i driver dei dispositivi
  - I driver per i dispositivi specifici presenti nel sistema di calcolo

Il sottosistema di I/O – 2

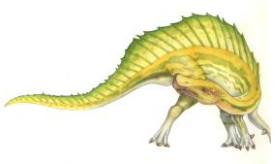
Spooling

Spooling is the process of sending output (intended for a printer or other peripheral) to a disk file, temporarily, so that it can be forwarded to the printer (or other peripheral) when it free.



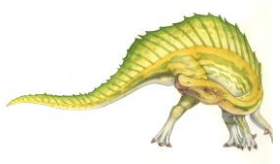
Protezione e sicurezza – 1

- Se diversi utenti usufruiscono dello stesso elaboratore che consente l'esecuzione simultanea e concorrente dei processi, l'accesso alle risorse del sistema di calcolo dovrà essere disciplinato da regole imposte dal SO
- ⇨ File, segmenti di memoria, CPU, altre risorse possono essere utilizzate solo dai processi che hanno ottenuto apposita autorizzazione dal SO
- **Protezione** — è l'insieme di tecniche usate per controllare l'accesso da parte di processi e/o utenti a risorse del sistema di calcolo
  - Le strategie di protezione devono fornire le specifiche dei controlli da attuare (politiche) e gli strumenti per la loro applicazione (meccanismi)



Protezione e sicurezza – 2

- Sicurezza — è il meccanismo di difesa implementato dal sistema per proteggersi da attacchi interni ed esterni
- Tipi di violazione
  - Denial–of–Service: malfunzionamento dovuto ad un attacco informatico in cui si esauriscono deliberatamente le risorse di un sistema di calcolo che fornisce un servizio, fino a renderlo non più in grado di erogarlo
  - Trojan: programmi che hanno una funzione conosciuta legittima e una funzione dannosa nascosta
  - Worm: malware in grado di autoreplicarsi
  - Virus: porzioni di codice dannoso che si legano ad altri programmi del sistema per diffondersi



Protezione e sicurezza – 3

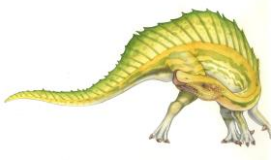
- Chi mette in atto gli attacchi alla sicurezza?
  - Hacker: *pirata informatico* che si impegna nell'affrontare sfide intellettuali per aggirare o superare creativamente le limitazioni di accesso ai sistemi
  - Cracker: chi si ingegna per eludere blocchi imposti da qualsiasi software al fine di trarne guadagno
  - Alcune tecniche di attacco sono simili, ma l'hacker è colui che sfrutta le proprie capacità informatiche per esplorare, divertirsi, apprendere, senza creare danni; al contrario il cracker è colui che sfrutta le proprie capacità (o, in certi casi, quelle degli altri) al fine di distruggere, ingannare e arricchirsi

Protezione e sicurezza – 4

- Più in dettaglio…
  - La funzione della **sicurezza** riguarda la prevenzione dall'uso illegale o dalle interferenze operate da persone o programmi fuori dal controllo del sistema operativo
  - La funzione di **protezione** riguarda l'accesso improprio a risorse del sistema causato da utenti accreditati
- In prima istanza, infatti, il sistema distingue i propri utenti, per determinare chi può fare cosa
  - L'identità utente (*user ID*) include nome dell'utente e numero associato — uno per ciascun utente
  - L'user ID garantisce l'associazione corretta di file e processi all'utente e ne regola la manipolazione
  - L'identificativo di gruppo permette inoltre ad un insieme di utenti di accedere correttamente ad un pool di risorse comuni (file e processi)

```
– es. ls -l pippo.c
-rw-r--r-- 1 susanna users 1064  Feb 6 2002 pippo.c
```

Protezione
r - permesso di lettura (directory, listing)
w- permesso di scrittura (directory, aggiungere file)
x - permesso di esecuzione (directory, accesso)



La virtualizzazione – 1

- La virtualizzazione è una tecnica che permette di eseguire un sistema operativo come applicazione all'interno di un altro SO
- I programmi di virtualizzazione sono software di emulazione
- L'emulazione viene utilizzata, per esempio, quando la CPU su cui è stata compilata un'applicazione è diversa da quella su cui verrà eseguita
  - Quando Apple è passata dalle CPU IBM Power alle Intel ha fornito l'emulatore Rosetta per permettere la portabilità delle vecchie applicazioni
  - Analogamente, Rosetta 2 emula le istruzioni Intel sui nuovi processori M1/M2 di Apple

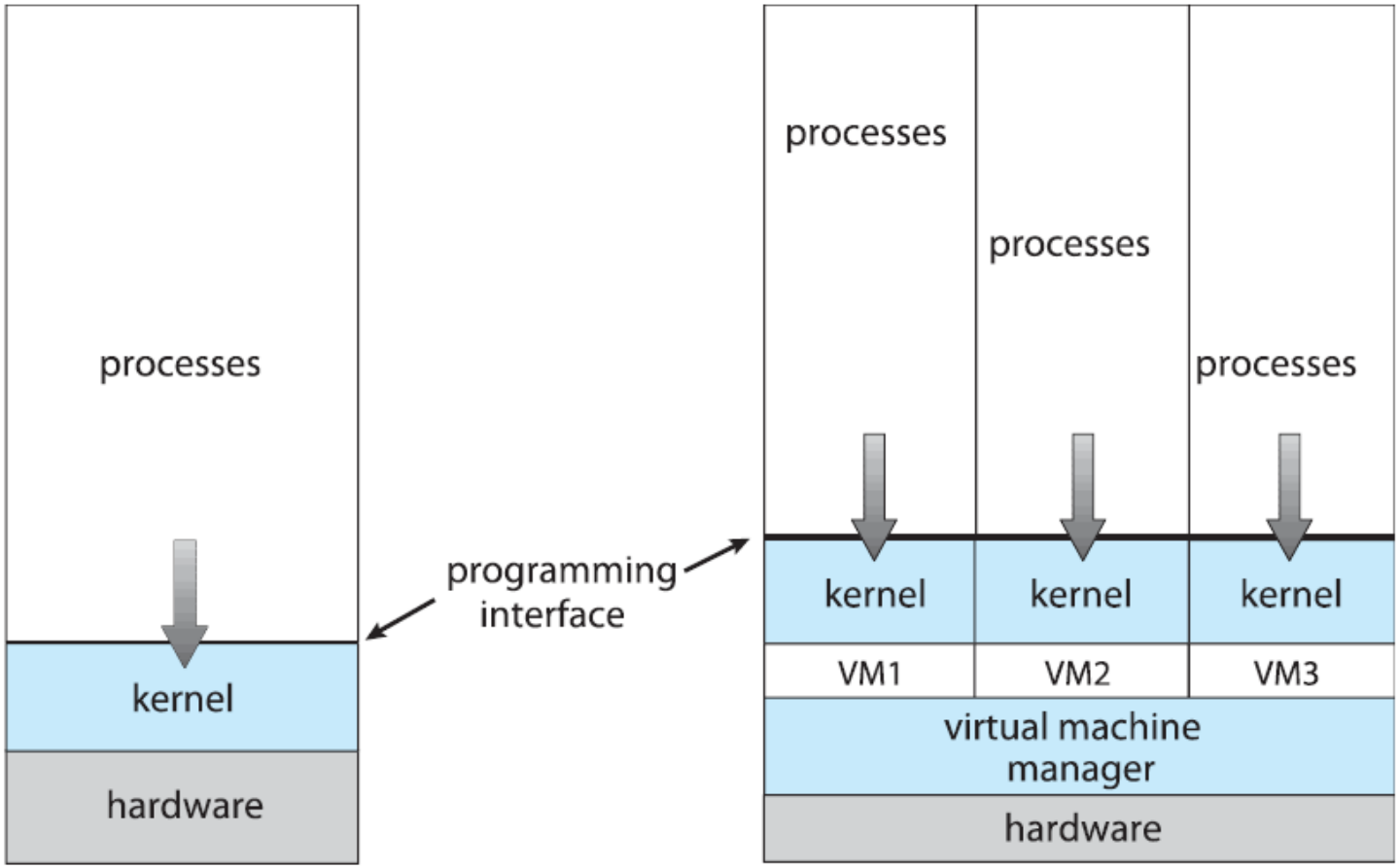
La virtualizzazione – 2

- Il concetto di emulazione può essere esteso per permettere ad un SO, scritto e compilato per una data piattaforma, di essere eseguito su piattaforme diverse
- Una macchina virtuale realizza un'interfaccia indistinguibile dalla macchina fisica sottostante: ogni processo ospite usufruisce di una copia virtuale del calcolatore
- Le risorse del computer fisico vengono condivise dall'hypervisor o VMM (*Virtual Machine Manager*) in modo che ciascuna macchina virtuale sembri possedere il proprio processore, la propria memoria e i propri dispositivi
- Sia l'hardware virtuale che il sistema operativo (ospite) vengono eseguiti in un ambiente (strato) isolato
- Le risorse del computer fisico vengono condivise in modo da creare le macchine virtuali

La virtualizzazione – 3

- L'hypervisor può essere parte di un SO host (per esempio un modulo) o un micro–kernel che genera le macchine virtuali (VM)
- L'hypervisor può sfruttare caratteristiche specifiche del processore (hardware virtualization)
  - Lo scheduling della CPU può creare l'illusione che ogni utente abbia un proprio processore
  - Lo spooling e il file system possono fornire dispositivi di I/O virtuali (per esempio, stampanti)
  - Lo spazio disco può essere "suddiviso" per creare dischi virtuali
  - Con due bit di modo, in certe architetture, si può avere una modalità esecutiva tipica dell'hypervisor (intermedia rispetto a modo utente e modo kernel)

La virtualizzazione – 4

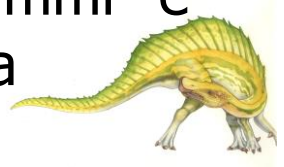


La virtualizzazione – 5

- Il concetto di macchina virtuale fornisce una protezione completa delle risorse di sistema (hardware e SO ospitante), dato che ciascuna macchina virtuale è isolata da tutte le altre
  - Questo isolamento, tuttavia, non permette, in generale, una condivisione diretta delle risorse
  - In Linux, condivisione possibile, per esempio, con *VirtIO*
- Per la condivisione di risorse…
  - Condivisione di un volume del file system
  - Rete di macchine virtuali: ogni macchina virtuale può inviare/ricevere informazioni sulla rete privata virtuale, modellata come una rete fisica, ma realizzata via software
  - La memoria inutilizzata viene ceduta ad altre VM

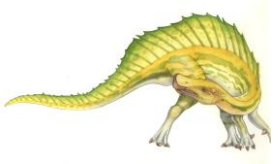
La virtualizzazione – 6

- Un sistema con macchine virtuali è un mezzo perfetto per la ricerca e lo sviluppo di sistemi operativi
  - Lo sviluppo del SO è effettuato sulla macchina virtuale, invece che sulla macchina fisica, così da non interferire con il normale funzionamento del sistema
  - Difficoltà nell'ottimizzare le prestazioni a causa dell'emulazione
- Ulteriori casi d'uso ed esempi
  - Computer Apple con Mac OS come host e Windows come guest
  - Sviluppo e messa a punto (quality assurance) di applicazioni per diversi SO sullo stesso sistema HW/SW
  - Gestione dei dati di varia provenienza nei data center
  - Possibilità di incrementare la vita dei programmi e mezzo per studiare vecchie architetture di sistema

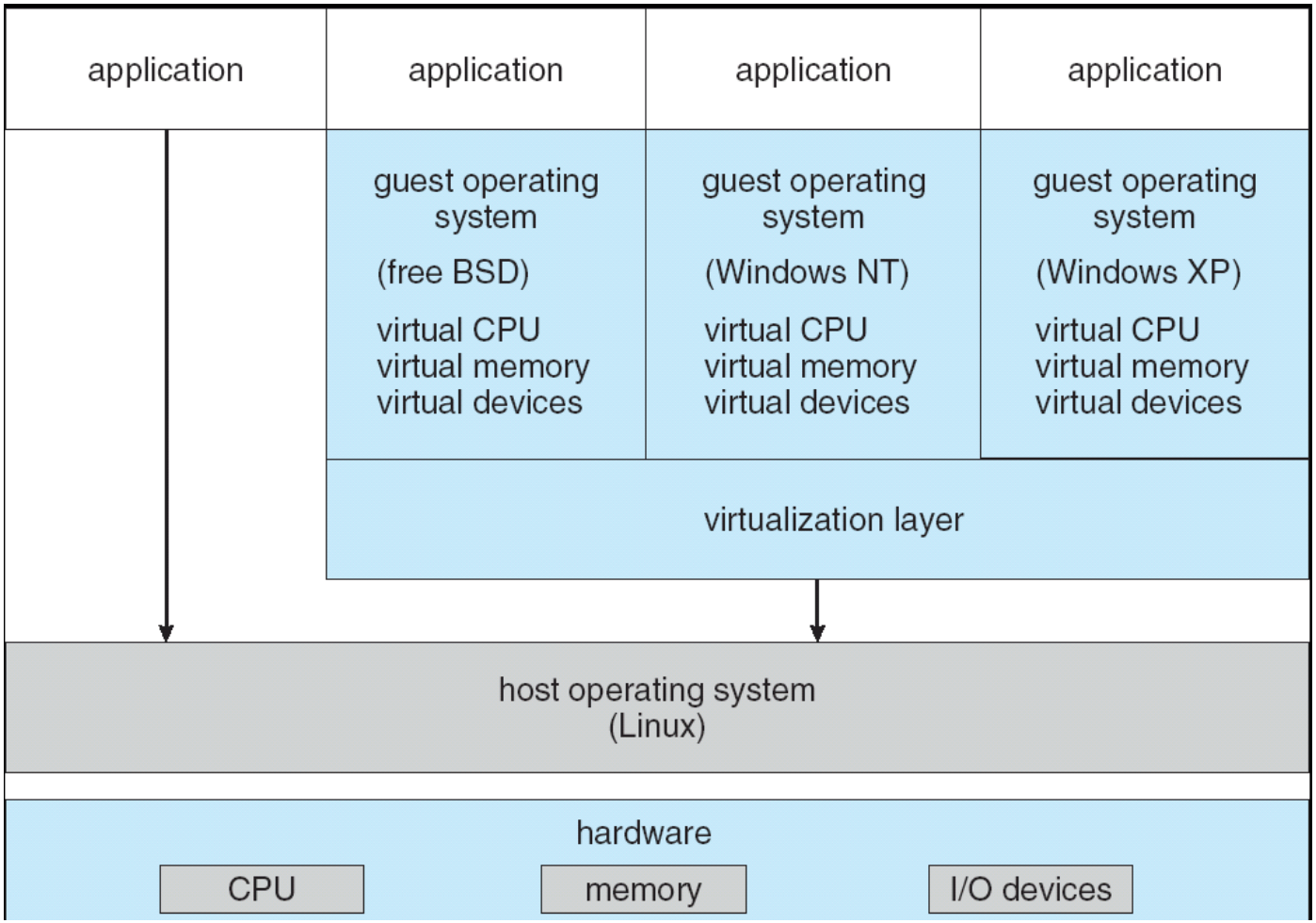


La virtualizzazione – 7

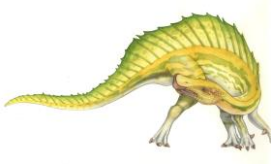
- Tra i vantaggi dell'utilizzo di macchine virtuali vi è inoltre il fatto di poter offrire contemporaneamente a utenti diversi ambienti operativi separati
- Il concetto di macchina virtuale è difficile da implementare per il notevole sforzo richiesto per fornire un duplicato esatto (sebbene meno potente) della macchina fisica
- Le più diffuse sono Kvm, VMware, Virtualbox, Parallels, ESXi, Citrix Hypervisor
- In alcuni casi i VMM possono fungere essi stessi da sistemi nativi, senza la presenza di un SO host



La virtualizzazione – 8



Architettura VMware



I vantaggi della virtualizzazione

Partizionamento

- Esecuzione di più sistemi operativi su una sola macchina fisica.
- Suddivisione delle risorse di sistema tra le macchine virtuali.

Isolamento

- Isolamento di guasti e problemi di sicurezza a livello di hardware.
- Protezione delle prestazioni grazie a controlli avanzati delle risorse.

Incapsulamento

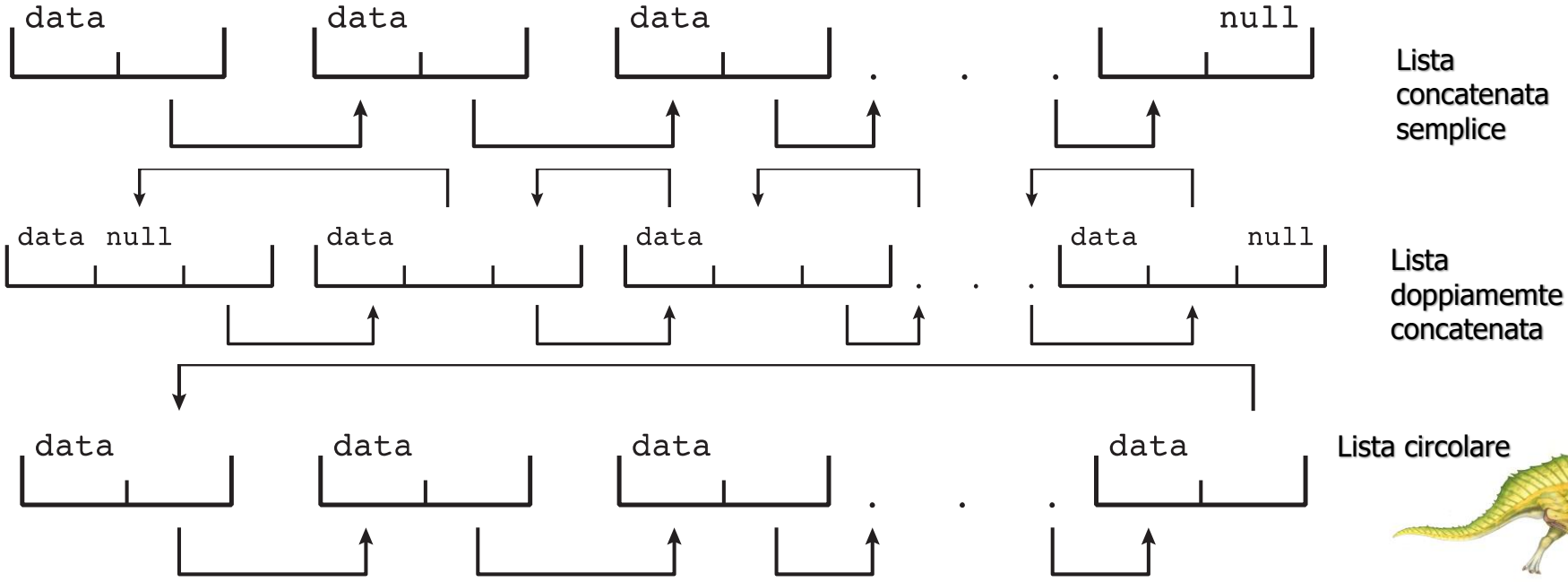
- Salvataggio su file dell'intero stato di una macchina virtuale.
- Spostamento e copia delle macchine virtuali con estrema facilità, in modo analogo ai file.

Indipendenza dall'hardware

- Provisioning o migrazione delle macchine virtuali a qualsiasi server fisico.

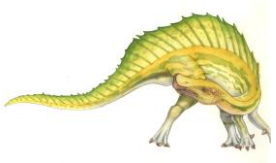
Strutture dati del kernel – 1

- Un array è una semplice struttura dati in cui ogni elemento è direttamente accessibile tramite un indice
  - La memoria principale è costruita come un array, così come il vettore degli interrupt
- Come procedere però in caso di dati di dimensione variabile? Come si può rimuovere un elemento, mantenendo l'ordine relativo degli altri?
  - Uso di liste concatenate



Strutture dati del kernel – 2

- Le liste concatenate possono contenere dati di diversa natura e dimensione (collezionati all'interno di record)
- Potenziale svantaggio: il costo della ricerca è $\mathcal{O}(n)$
- Le liste sono talvolta utilizzate direttamente dagli algoritmi del kernel, ad esempio in per certe modalità di scheduling della CPU o nell'algoritmo Clock per la sostituzione delle pagine
- Più frequentemente le si utilizza per implementare pile e code

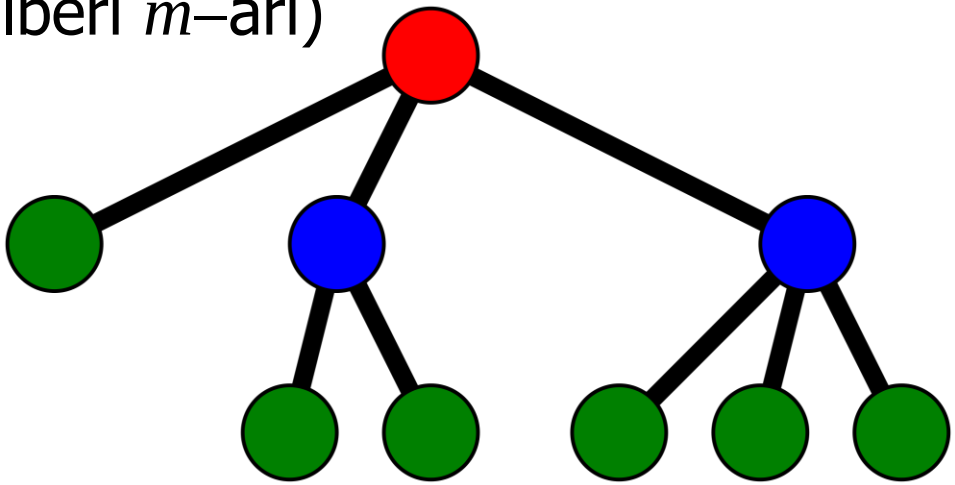


Strutture dati del kernel – 3

- Lo stack viene acceduto con politica LIFO (Last In First Out) e le operazioni di inserimento ed estrazione (cancellazione) vengono rispettivamente dette *push* e *pop*
  - Il SO utilizza lo stack per gestire le chiamate di funzione
    - Al momento della chiamata, si inseriscono nello stack l'indirizzo di ritorno, le variabili locali e i parametri attuali
    - …che vengono recuperati dalla funzione chiamata
  - Anche il cambio di contesto viene gestito tramite stack
- Le code si accedono in modalità FIFO (First In First Out)
  - Il SO gestisce come una coda, per esempio, i documenti inviati ad una stampante o i processi in attesa di ottenere l'accesso alla CPU

Strutture dati del kernel – 4

- Gli alberi sono utilizzati per organizzare i dati in maniera gerarchica e sono basati sulla relazione causale padre–figlio
- Gli alberi possono avere diversa *arietà*, cioè, da ciascun nodo, possono dipartirsi al più un numero m di archi (alberi m–ari)

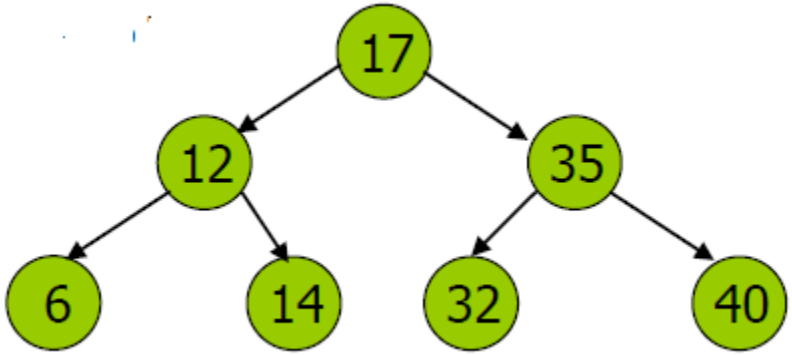


- Se ogni padre ha, al più, due figli l'albero si dice binario

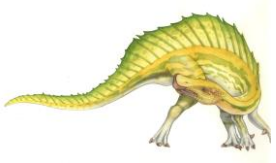
Strutture dati del kernel – 5

- Sono alberi binari di ricerca quelli per cui vale la relazione d'ordine figlio$_{sinistro}$ < padre < figlio$_{destro}$
  - Nel caso peggiore la ricerca costa $\mathcal{O}(n)$, ma, a patto di mantenere l'albero "bilanciato" il costo medio di questa operazione scende a $\mathcal{O}(\log_2 n)$

    {17,35,12,14,32,40,6}



  - Linux utilizza un albero binario di ricerca bilanciato nel suo algoritmo di scheduling della CPU



Strutture dati del kernel – 6

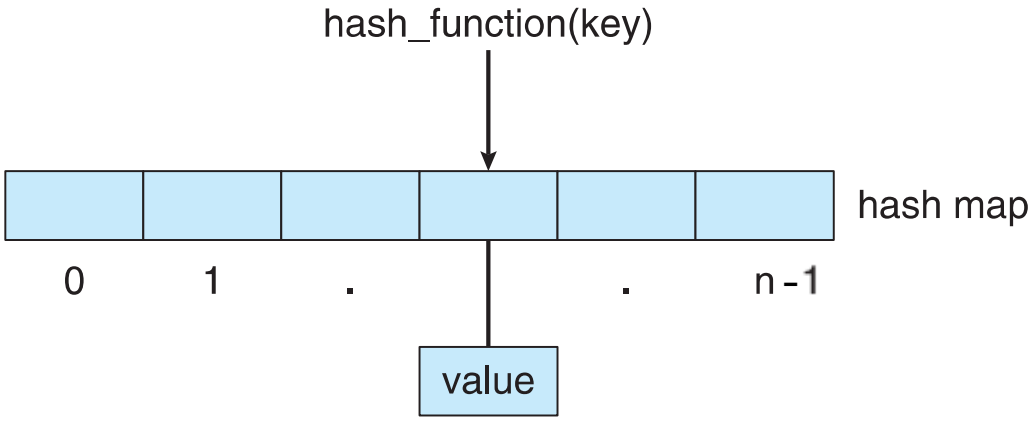
- **Problema** — Memorizzare in maniera opportuna un insieme di dati, tipicamente sotto forma di record, in modo da poter reperire un qualsiasi elemento dell'insieme con un numero "piccolo" di tentativi
  - Cosa significa "piccolo" ?
    - Indipendente (o quasi) dalla dimensione della tabella su cui si effettua la ricerca, quindi con una complessità in tempo pari a $\mathcal{O}(1)$
- **Funzioni hash**
  - h: $K \rightarrow \{0, 1, 2, \ldots, n-1\}$ (*to hash*, in inglese, significa sminuzzare)
    - K: insieme dei valori distinti che possono essere assunti dalle chiavi dei record
    - n: dimensione del vettore in cui si intende memorizzare la tabella

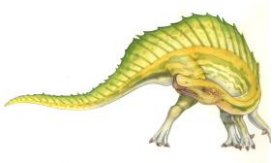
Strutture dati del kernel – 7

- Ipotesi: K sottoinsieme dei numeri naturali
- ⇨ Possibile funzione di accesso:

$$h(k) = k \text{ MOD } n,\ k \in K$$

  - Valore della funzione sempre compreso fra 0 e $n-1$



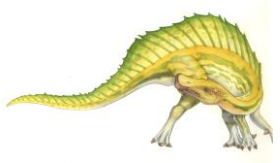


Strutture dati del kernel – 8

- Se K non è un sottoinsieme dei numeri naturali
  - Esempio: insieme di stringhe alfanumeriche
  - La funzione hash si applica a numeri

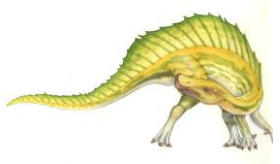
⇨ Per utilizzarla in corrispondenza di una chiave non numerica occorre associare alla chiave un valore numerico

- Necessità di definire funzioni hash generali
  - Associazione di un valore numerico ad una chiave di qualunque tipo
  - Applicazione della funzione hash a tale valore
  - Esempio: si utilizza la somma dei codici ASCII dei caratteri che costituiscono la stringa



Strutture dati del kernel – 9

- **Collisione:** associazione, da parte di una trasformazione hash, della stessa posizione a chiavi distinte
- *Le chiavi sono dette* ***sinonime***
- **Esempio:**
  [10,12,20,23,27,30,31,39,42,44,45,49,53,57,60]
  - h(chiave) = chiave MOD 15
    - Posizione 0 ← 30, 45, 60
    - Posizione 8 ← 23, 53
    - Posizione 12 ← 12, 27, 42, 57
- Ciascuna posizione dell'array può contenere al più un elemento; occorre:
  - Ridurre il più possibile le collisioni
  - Gestirle quando si verificano



Strutture dati del kernel – 10

- Funzioni di *hashing perfetto* (che evitano i duplicati) sono difficili da trovare anche per tabelle grandi — cioè accettando un notevole spreco di memoria
- Esempio: *paradosso del compleanno*
  Dato un gruppo di 23 persone, ci sono più del 50% di probabilità che due di esse siano nate nello stesso giorno dell'anno
  - ⇨ In altre parole, se scegliamo una funzione aleatoria che trasforma 23 chiavi in un indirizzo di una tabella di 365 elementi, la probabilità che due chiavi NON collidano è solo 0.4927 (meno della metà)
- Individuare una funzione di accesso che porti ad un numero ridotto di collisioni è un problema complesso

Strutture dati del kernel – 11

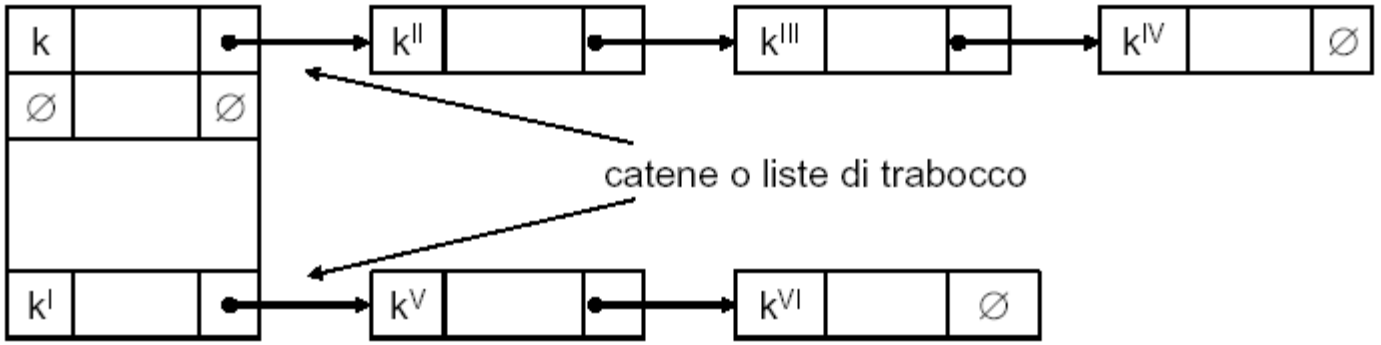
- Tuttavia… numero di collisioni ridotto drasticamente se accettiamo uno spreco del 20% di memoria extra
- Esempio: array di 19 elementi (indicizzati da 0 a 18)
  - Posizione 0 ← 57
  - Posizione 8 ← 27
  - Posizione 1 ← 20, 39
  - Posizione 10 ← 10
  - Posizione 3 ← 60
  - Posizione 11 ← 30, 49
  - Posizione 4 ← 23, 42
  - Posizione 12 ← 12, 31
  - Posizione 6 ← 44
  - Posizione 15 ← 53
  - Posizione 7 ← 45

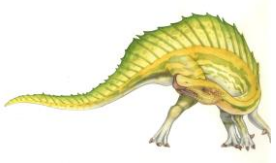
0	57
1	20, 39
2	
3	60
4	23, 42
5	
6	44
7	45
8	27
9	
10	10
11	30, 49
12	12, 31
13	
14	
15	53
16	
17	
18	

- Collisioni non eliminate del tutto

Strutture dati del kernel – 12

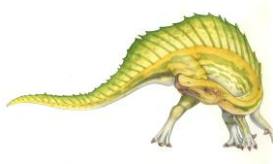
- Uso di liste concatenate destinate alla memorizzazione degli elementi che, in inserimento, hanno portato ad una collisione
- Ricerca di un elemento di chiave k
  - Si calcola $h(k)$
    - Se si verifica una collisione allora si accede alla lista associata alla posizione $h(k)$ e la si scandisce





Strutture dati del kernel – 13

- Il costo dell'operazione di ricerca — realizzata in modo lineare relativamente alle liste di elementi in collisione — si mantiene pressoché indipendente da n (numero degli elementi contenuti nella tabella)
- Nei sistemi operativi, per l'autenticazione si procede come segue:
  - L'utente inserisce username e password
  - Viene applicata la funzione hash allo username (cui corrisponde un ID utente numerico) in modo da recuperare la password memorizzata in tabella
  - La password viene confrontata con quella inserita dall'utente per l'autenticazione



Strutture dati del kernel – 14

- Una bitmap è una stringa di n caratteri binari utilizzabile per rappresentare lo stato di n elementi

  001011101…

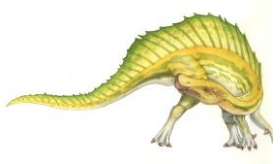
- Efficienza in termini di spazio utilizzato
- Nei sistemi operativi, la gestione dello spazio sui primi dischi magnetici offriva un ottimo esempio di utilizzo di bitmap
  - Un partizione di disco di medie dimensioni può essere divisa in diverse migliaia di unità, dette blocchi (dim. media: 4KB)
  - Si utilizza una bitmap per indicare la disponibilità o meno di ognuno di essi

Ambienti di calcolo – 1

- Elaborazione tradizionale
  - Con l'evoluzione tecnologica, i confini tra i diversi ambienti di elaborazione tradizionali diventano sempre più sfumati
  - Negli uffici…
    - Fino a non molti anni fa: diversi PC connessi in rete, con server per servizi di accesso ai file e per stampa
    - Ora: *portali* che permettono l'accesso alla rete Internet e a risorse allocate su server interni e sistemi remoti tramite dispositivi mobili (e *thin client*)
    - Facilità di sincronizzazione con i dispositivi mobili per mezzo di reti wireless
  - Nelle case…
    - Fino a non molti anni fa: un PC connesso alla rete via modem
    - Ora: piccole reti locali, connessioni veloci a Internet, barriere anti–intrusione (*firewall*)

Ambienti di calcolo – 2

- Mobile computing
  - Ad oggi, le funzionalità disponibili sui dispositivi mobili si sono talmente arricchite da rendere indistinguibili computer portatili e tablet
  - Talvolta le funzioni di un moderno dispositivo mobile non sono disponibili, o non sono facilmente realizzabili, su computer desktop o portatili
    - Posta elettronica ed accesso al Web
    - Accesso/creazione a/di dati multimediali (musica, foto e video in alta definizione, ebook)
    - Presenza di GPS, accelerometro e giroscopio: rotazione dello schermo, navigazione, giochi, augmented reality
  - Accesso ai servizi online per mezzo dello standard wireless IEEE 802.11 o tramite reti cellulari



Ambienti di calcolo – 3

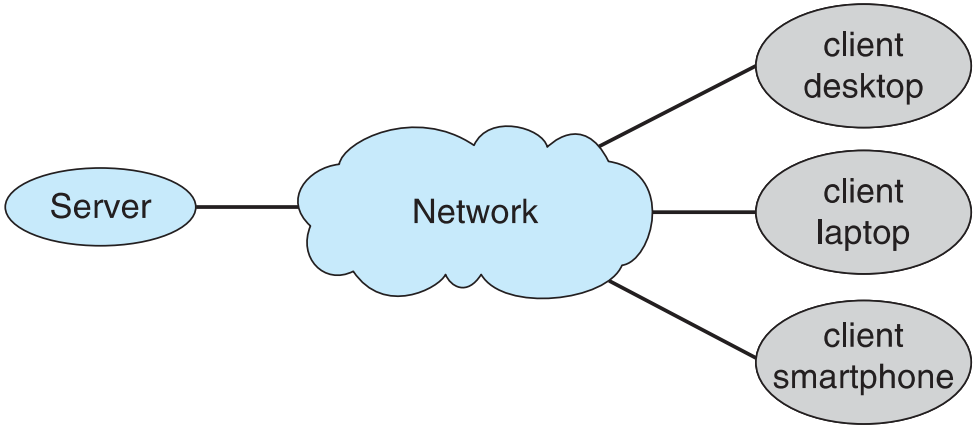
- Mobile computing
  - Anche la capacità di memoria e la velocità dei processori si stanno sempre più avvicinando a quelle dei portatili, anche se si mantiene l'attenzione al risparmio energetico
    - Esempio: Nell'iPhone 14 Pro, processore A16 Bionic con CPU con 6 core — 2 power core Avalanche a 3.46 GHz e 4 efficiency core Blizzard a 2.02 GHz —, GPU con 5 core, Neural Engine con 16 core, 6 GB di RAM, 1 TB di memoria
  - Mercato dei SO dominato da Apple iOS e Google Android

Ambienti di calcolo – 4

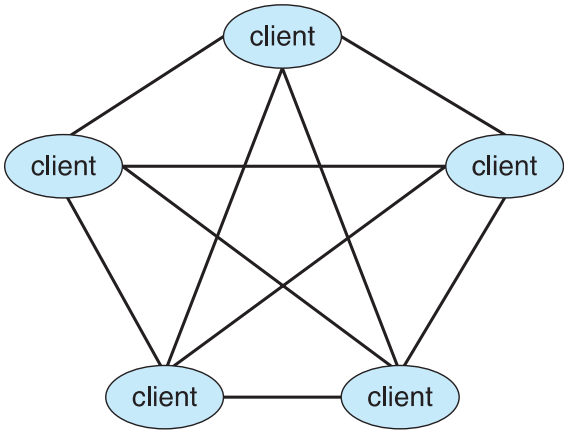
- Sistemi distribuiti
  - Un sistema distribuito è un insieme (eterogeneo) di calcolatori che non condividono né la memoria né il clock; ciascun processore ha la sua propria memoria locale
  - I computer nel sistema sono connessi attraverso una rete di comunicazione
    - Wide Area Network (WAN)
    - Metropolitan Area Network (MAN)
    - Local Area Network (LAN)
    - Personal Area Network (PAN)
  - La comunicazione avviene secondo un dato *protocollo* (TCP/IP è il più diffuso)
  - Un sistema distribuito fornisce agli utenti l'accesso a varie risorse di sistema
  - L'accesso a risorse condivise consente di:
    - Accelerare l'elaborazione
    - Aumentare la disponibilità di dati
    - Migliorare l'affidabilità

Ambienti di calcolo – 5

- Modello client–server
  - Terminali e mainframe soppiantati da PC e server
  - I PC fungono da *client* e richiedono servizi a server
  - I *server* permettono l'accesso a servizi e risorse (i.e., stampa, memorizzazione e reperimento di file, database, accesso alla rete Internet)
    - Server elaborativi — per esempio, un server che ospita una base di dati e risponde a interrogazioni su di essa
    - File server — per esempio, un server Web che fornisce pagine ai browser dei client



Ambienti di calcolo – 6



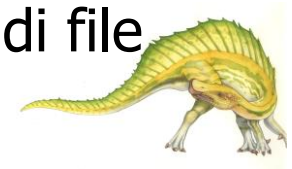
- Modello peer–to–peer
  - Un modello diverso di sistema distribuito
  - P2P non distingue client e server, ma è costituita da *nodi equivalenti* (*peer*, appunto) che fungono sia da client che da server verso altri nodi della rete
  - Ogni nodo deve connettersi ad una rete P2P…
    - …registrando i propri servizi in un registro centralizzato di consultazione della rete
    - o inoltrando le proprie richieste di servizio a tutti gli altri nodi in broadcast (utilizzando un protocollo di scoperta)
  - Esempi storici sono Napster (registro centralizzato) e Gnutella (protocollo di scoperta) e, più recentemente, i servizi Voice over IP (VoIP) come Skype

Cloud computing – 1

- La diffusione della connettività Internet, in mobilità e wi–fi, e il numero sempre maggiore di smartphone e tablet hanno contribuito al recente successo dei servizi di cloud computing
- L'espressione, che in italiano potrebbe tradursi come "nuvola informatica", si riferisce alle piattaforme e alle tecnologie che permettono di archiviare file (documenti, immagini, video, musica) o di sviluppare/utilizzare programmi e applicazioni direttamente sui server di chi fornisce il servizio, anziché sul proprio dispositivo

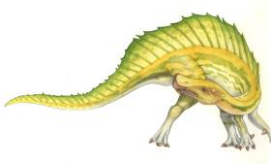
Cloud computing – 2

- Pertanto, il cloud computing è una tecnica che permette la fruizione di risorse computazionali, di storage e di applicazioni come servizi di rete
  - Non si conservano più i file direttamente su memorie locali, ma su server remoti
  - Ugualmente dicasi per i programmi: anziché installarli, possono essere usati direttamente online
- È un'estensione logica della virtualizzazione perché utilizza la virtualizzazione come strumento base per offrire le proprie funzionalità
  - EC2 (*Elastic Compute Cloud*) di Amazon dispone di migliaia di server, milioni di macchine virtuali e petabyte di spazio dati accessibili via Internet
  - Google Drive è la suite Office dell'azienda di Mountain View, permette l'editing e l'archiviazione di vari tipi di file
  - Dropbox, OneDrive, iCloud…



Cloud computing – 3

- Tipologie di cloud computing
  - Cloud pubblico: disponibile attraverso Internet a chiunque si abboni al servizio
  - Cloud privato: gestito da un'azienda ad uso interno
  - Cloud ibrido: comprende componenti sia pubbliche che private

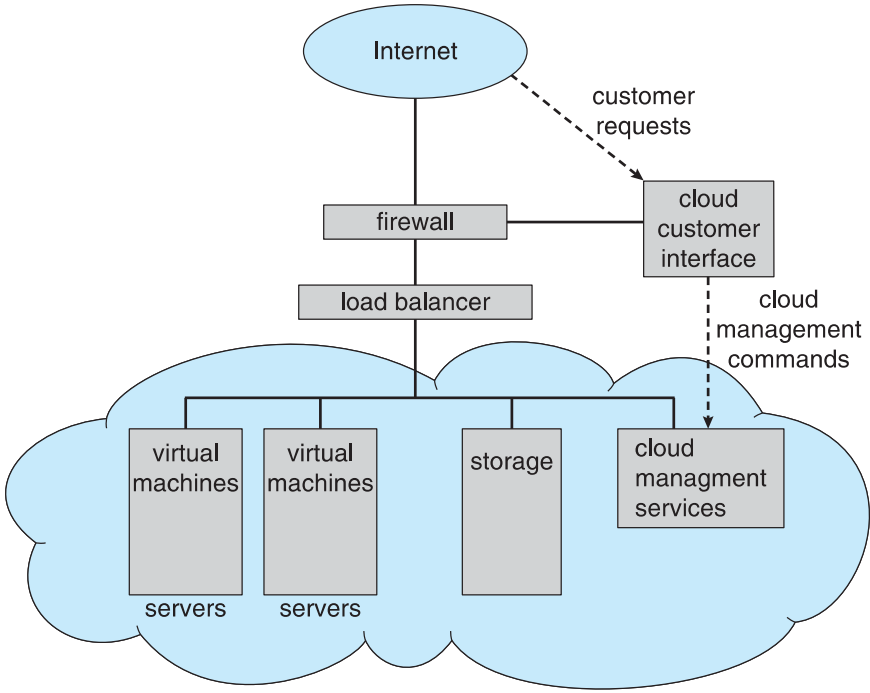


Cloud computing – 4

- Tipologie di servizi
  - SaaS (*Software as a Service*): una o più applicazioni fruibili via Internet (es.: word processor, fogli di calcolo)
  - Daas (*Data as a Service*): disponibilità di dati, ai quali gli utenti possono accedere tramite qualsiasi applicazione, come se fossero residenti su un disco locale
  - Haas (*Hardware as a Service*): computer messi a disposizione a utenti che inviano dati e ricevono risultati
  - PaaS (*Platform as a Service*): ambiente software predisposto per usi applicativi via Internet (es.: database server)
  - IaaS (*Infrastructure as a Service*): server o storage disponibili attraverso Internet (es.: dispositivi per backup)

Cloud computing – 5

- Gli ambienti di cloud computing inglobano SO tradizionali, VMM per le macchine virtuali, oltre agli strumenti specifici per la gestione del cloud
  - Vmware vCloud Director
  - Eucalyptus (open–source)
- Inoltre…
  - La connettività Internet impone l'uso intensivo di dispositivi di sicurezza (firewall) e di bilanciatori di carico per lo smistamento efficiente delle applicazioni



Sistemi embedded – 1

- Rappresentano la tipologia dominante di elaboratore: sono ubiquitari (elettrodomestici, motori delle auto, robot industriali,…)
- Normalmente installati su sistemi rudimentali, hanno compiti molto precisi e offrono funzionalità limitate e interfaccia utente scarsamente sviluppata (se presente)
- I sistemi embedded sono caratterizzati da una grande variabilità
  - Elaboratori general–purpose con sistemi operativi standard che sfruttano applicazioni create appositamente per implementare una particolare funzionalità
  - Microprocessori dedicati con SO special–purpose
  - Application Specific Integrated Circuit (ASIC) privi di SO

Sistemi embedded – 2

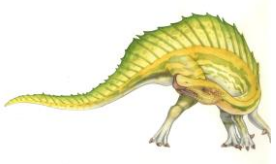
- La tipologia più diffusa per gli embedded sono i sistemi operativi real–time
  - Vincoli definiti a priori sui tempi di esecuzione
  - L'elaborazione si considera portata a termine correttamente solo se il compito viene eseguito entro i limiti prestabiliti (**hard** real–time)
  - In effetti, i vincoli sui tempi di risposta sono così stringenti che, se non vengono rispettati, il sistema di controllo è inutile se non dannoso
  - Esempi:
    - Programmi di controllo delle superfici di volo di un aereo
    - Robot per la costruzione di autovetture

Sistemi operativi open–source – 1

- Disponibili in formato sorgente anziché come codice binario compilato
- Vantaggi
  - Non è necessario effettuare il *reverse engineering* per comprendere il funzionamento del sistema
    - Operazione che non consente, ad esempio, di ricostruire i commenti integrati al codice
  - Costituzione di una comunità di programmatori e aziende che contribuiscono allo sviluppo, al debugging, all'assistenza e al supporto (spesso gratuito) agli utenti
  - Codice più sicuro e bug scoperti e risolti velocemente
    - Molti più «occhi puntati sul codice»
  - Libertà

Sistemi operativi open–source – 2

- Qualche problema…
  - …con la gestione dei diritti digitali (*DRM – Digital Rights Management*): limiti imposti facilmente violabili, modificando il codice aperto
  - Le leggi di molti paesi, incluso il *Digital Millennium Copyright Act* (DMCA) in USA, rendono illegale recuperare un codice DRM per provare a eludere la protezione di materiale digitale
  - I SO chiusi (e altri software) possono invece implementare blocchi di accesso a contenuti protetti da copyright



Sistemi operativi open–source – 3

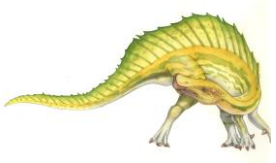
- Come nascono…
  - 1983: Richard Stallman da vita al progetto GNU (è un acronimo ricorsivo che significa GNU is Not Unix), con la finalità di creare un SO gratuito, open–source e compatibile con UNIX
  - Copyleft (1984)
    - Il concetto di *Copyleft* nacque quando Richard Stallman stava lavorando ad un interprete Lisp
    - L'azienda Symbolics chiese di poter utilizzare l'interprete e Stallman accettò di fornire una versione di pubblico dominio
    - Symbolics estese e migliorò l'interprete Lisp, ma quando Stallman volle accedere ai miglioramenti, Symbolics rifiutò

Sistemi operativi open–source – 4

- L'espressione inglese copyleft è un gioco di parole sul termine *copyright* nel quale la parola "right" significa "diritto" (in senso legale), ma giocando sul suo secondo significato (ovvero "destra") viene scambiata con "left" ("sinistra", ma anche "lasciato")
- Il copyleft è il "permesso d'autore": individua un modello di gestione dei diritti d'autore basato su un sistema di licenze attraverso le quali l'autore (in quanto detentore originario dei diritti) indica ai fruitori dell'opera che essa può essere utilizzata, diffusa e modificata liberamente, nel rispetto di alcune condizioni essenziali
- Nella versione originaria del copyleft, la condizione principale obbliga i fruitori dell'opera, nel caso vogliano distribuire l'opera modificata, a farlo sotto lo stesso regime giuridico (e generalmente sotto la stessa licenza)
- In questo modo, il regime di copyleft e tutto l'insieme di libertà da esso derivanti sono sempre garantiti

Sistemi operativi open–source – 5

- 1985: Viene pubblicato il Manifesto GNU che sostiene che tutti i software dovrebbero essere gratuiti ed open–source; si costituisce la FSF, Free Software Foundation, con lo scopo di incoraggiare il libero scambio di sorgenti ed il libero utilizzo del software
  - La FSF distribuisce il software con licenza GPL (General Public License)
  - GPL presuppone la distribuzione congiunta di sorgente e binario ed impone che qualsiasi cambiamento apportato al sorgente sia reso disponibile con licenza GPL
  - Anche la licenza Creative Commons "Attribution Sharelike" è una licenza copyleft



Sistemi operativi open–source – 6

- 1991: Linus Torvalds rilascia un kernel rudimentale, simile a UNIX, utilizzando compilatori, librerie, strumenti di GNU – nasce LINUX (versione 1.0, 13 marzo 1994)
  - Moltissime distribuzioni: Red Hat, SUSE, Fedora, Debian, Slackware, Ubuntu, Sabayon, etc.
  - L'ultima versione del kernel Linux è la 6.2.5 ed è stata rilasciata l'11 marzo 2023 (5.0 rilasciata il 4 marzo 2019, 6.0 rilasciata il 4 ottobre 2022)

"Il cambio della numerazione da 4.x a 5 non implica che è in arrivo qualcosa di speciale. Se volete avere un motivo ufficiale diciamo che ho finito le dita delle mani e dei piedi per contare, così il kernel 4.21 è diventato 5.0"

Sistemi operativi open–source – 7

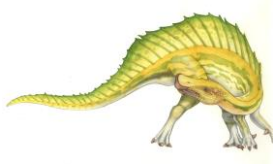
- Attualmente: il progetto GNU ha prodotto molti strumenti compatibili con UNIX, quali compilatori, editor e varie utility, ma non ha mai prodotto (e rilasciato) un kernel «definitivo»
  - Hurd, il kernel GNU, è in sviluppo, ma è ancora lontano dall'essere pronto all'uso quotidiano
  - Utility e programmi di sistema GNU utilizzati con kernel LINUX: GNU/LINUX
  - Debian GNU/Hurd 2021 Bullseye, 14.08.2021

"The latest releases are GNU Hurd 0.9, GNU Mach 1.8, GNU MIG 1.8, 18.12.2016. The Hurd is developed by a few volunteers in their spare time. The project welcomes any assistance you can provide. Porting and development expertise is still badly needed in many key areas."

Sistemi operativi open–source – 8

- Il software libero e il software open–source sono due idee "filosoficamente" distinte, sostenute da gruppi di persone diverse

  https://www.gnu.org/philosophy/philosophy.en.html
  - Il software libero non solo rende il codice sorgente disponibile, ma è anche dotato di una licenza che ne consente l'uso, la ridistribuzione e la modifica senza costi
  - Il software open–source non offre necessariamente tale licenza
- Linux è il SO open–source più famoso, con alcune distribuzioni libere e altre solo open–source
- Windows è proprietario e closed–source
- Mac OS è ibrido: contiene il kernel Darwin open–source, ma include anche componenti chiuse e proprietarie



Sistemi operativi open–source – 9

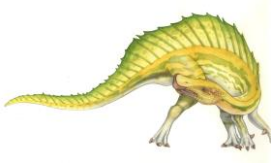
- UNIX BSD
  - Derivato da UNIX di AT&T, risale al 1978, e le sue prime versioni furono rilasciate dalla UCB, in codice sorgente e in binario, ma non open–source (necessaria la licenza AT&T)
  - Sviluppo rallentato dalla querela di AT&T: versione completa open–source del sistema (4.4BSD–lite) nel 1994
  - Varie distribuzioni:
    - FreeBSD, NetBSD, OpenBSD, DragonflyBSD
  - Darwin, il componente kernel fondamentale di Mac OS e di iOS, è basato su UNIX BSD ed è open–source (consultare https://opensource.apple.com/)

Sistemi operativi open–source – 10

- Solaris
  - Sviluppato, a partire dal 1991, da Sun Microsystems come derivato di UNIX System V di AT&T
  - Nel 2005 la Sun, nell'ambito del progetto OpenSolaris, rese disponibile parte del SO, continuando costantemente ad evolvere il codice open–source
  - Ad oggi, tuttavia, parte del codice è ancora di proprietà AT&T e, dal 2009, con l'acquisto di Sun da parte di Oracle, non vi sono stati ulteriori sviluppi "ufficiali"
  - Solaris può comunque essere compilato da sorgente e collegato (tramite linker) al codice binario delle componenti a codice chiuso
  - Illumos, portato avanti da gruppi di sviluppatori liberi sulla base di Solaris, ne ha comunque esteso le funzionalità

Impatto didattico dell'open–source – 1

- Il movimento a sostegno dei software gratuiti spinge legioni di programmatori a creare migliaia di progetti open–source, correlati anche allo sviluppo di SO
- Ultimo aspetto etico legato al software libero è quello che i progetti open–source permettono di…
  - …utilizzare il codice sorgente come strumento di apprendimento, con l'opportunità di modificarlo, testar-lo, contribuire all'individuazione/eliminazione di errori, customizzarlo
  - …esplorare SO già maturi e completi, con i relativi compilatori, programmi di sistema, interfacce utente, applicazioni



Impatto didattico dell'open–source – 2

There has never been a more interesting time to study operating systems, and it has never been easier. The open-source movement has overtaken operating systems, causing many of them to be made available in both source and binary (executable) format. The list of operating systems available in both formats includes Linux, BSD UNIX, Solaris, and part of Mac OS. The availability of source code allows us to study operating systems from the inside out. Questions that we could once answer only by looking at documentation or the behavior of an operating system we can now answer by examining the code itself.

Operating systems that are no longer commercially viable have been open-sourced as well, enabling us to study how systems operated in a time of fewer CPU, memory, and storage resources. An extensive but incomplete list of open-source operating system projects is available from https://curlie.org/Computers/Software/Operating_Systems/Open_Source/

In addition, the rise of virtualization as a mainstream (and frequently free) computer function makes it possible to run many operating systems on top of one core system. For example, VMware (http://www.vmware.com) provides a free "player" for Windows on which hundreds of free "virtual appliances" can run. Virtualbox (http://www.virtualbox.com) provides a free, open-source virtual machine manager on many operating systems. Using such tools, students can try out hundreds of operating systems without dedicated hardware.

The advent of open-source operating systems has also made it easier to make the move from student to operating-system developer. With some knowledge, some effort, and an Internet connection, a student can even create a new operating system distribution. Just a few years ago, it was difficult or impossible to get access to source code. Now, such access is limited only by how much interest, time, and disk space a student has.

Suggerimenti

- Per eseguire Linux su un altro sistema operativo:
  - Scaricare il software di virtualizzazione gratuito Virtual-box VMM tool all'indirizzo www.virtualbox.org e installar-lo sul proprio sistema
  - Scegliere da virtualboxes.org/images/ un'immagine del sistema operativo preconfigurata
  - Avviare la macchina virtuale all'interno di Virtualbox

Fine del Capitolo 1