Capitolo 1

Introduzione

OBIETTIVI

- **Panoramica dei più importanti componenti di un sistema operativo.**
- **Organizzazione degli elementi essenziali di un elaboratore.**

Un **sistema operativo** è un insieme di programmi (*software*) che gestisce gli elementi fisici di un calcolatore (*hardware*); fornisce una piattaforma ai programmi applicativi e agisce da intermediario fra l'utente e la struttura fisica del calcolatore. Un aspetto sorprendente dei sistemi operativi è quanto siano diversi i modi in cui eseguono questi compiti: i sistemi operativi per i *mainframe* si progettano innanzi tutto per ottimizzare l'utilizzo delle risorse; i sistemi operativi per PC consentono l'esecuzione di un'ampia varietà di programmi, dai giochi ai programmi gestionali; quelli per i sistemi palmari si progettano per fornire un ambiente in cui l'utente possa interagire facilmente col calcolatore per l'esecuzione dei programmi. Alcuni sistemi operativi si progettano per essere d'*uso agevole*, altri per essere *efficienti* e altri ancora per possedere una combinazione di tali qualità.

Prima di esplorare i particolari del funzionamento di un calcolatore, contempleremo brevemente la struttura del sistema, a partire dalle funzioni basilari connesse ad avvio, ingresso, uscita (I/O, per Input/Output) e memorizzazione dei dati. Inoltre, delineeremo l'architettura essenziale dell'elaboratore, grazie alla quale si può scrivere un sistema operativo funzionale.

In ragione della sua complessità e ampiezza, un sistema operativo deve essere costruito gradualmente, per parti. Ciascuna di loro dovrebbe rappresentare un'unità ben riconoscibile del sistema, dotata di funzioni, dati in entrata e in uscita accuratamente definiti. Questo capitolo presenta una panoramica generale dei principali componenti di un sistema operativo.

1.1 Che cos'è un sistema operativo

La nostra analisi parte dalla considerazione del ruolo del sistema operativo nell'insieme dei sistemi di calcolo. Un sistema di calcolo si può suddividere in quattro componenti: *dispositivi fisici*, *sistema operativo*, *programmi applicativi* e *utenti* (Figura 1.1).

I **dispositivi fisici** composti dall'unità centrale d'elaborazione, cioè la CPU (*central processing unit*), dalla **memoria** e dai dispositivi d'immissione ed emissione dei dati, cioè l'I/O (*input/output*), forniscono al sistema le risorse di calcolo fondamentali. I **programmi appli-**

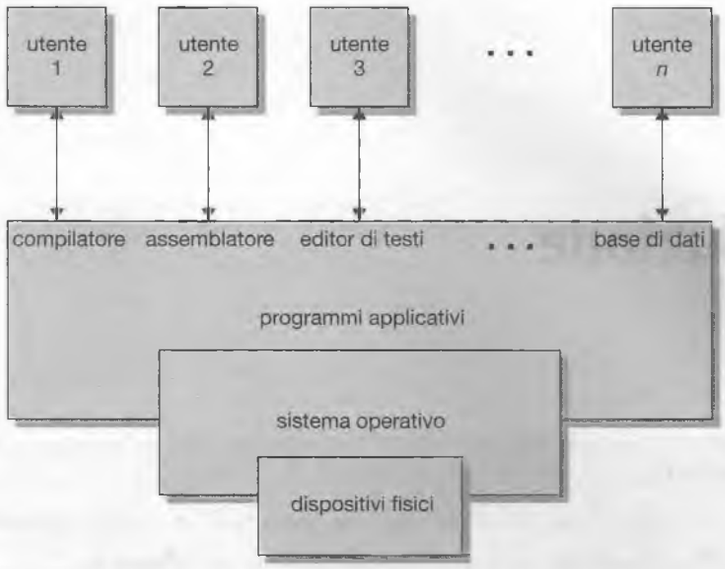

Figura 1.1 Componenti di un sistema di calcolo.

cativi (editor di testo, fogli di calcolo, compilatori e programmi di consultazione del Web) definiscono il modo in cui si usano queste risorse per la risoluzione dei problemi computazionali degli utenti. Il sistema operativo controlla e coordina l'uso dei dispositivi da parte dei programmi applicativi per gli utenti.

Un sistema di calcolo si può anche considerare come l'insieme dei suoi elementi fisici, programmi e dati. Il sistema operativo offre gli strumenti per impiegare in modo corretto queste risorse; non compie operazioni di per sé utili, ma definisce semplicemente un *ambiente* nel quale altri programmi possono lavorare in modo utile.

Per un maggior approfondimento del ruolo del sistema operativo, lo esploriamo da due punti di vista: quello degli utenti e quello del sistema.

1.1.1 Punto di vista dell'utente

La percezione di un calcolatore da parte di un utente dipende principalmente dall'interfaccia impiegata. La maggior parte degli utenti usa PC, composti da schermo, tastiera, mouse e un'unità di sistema. I PC sono progettati per un singolo utente, che impiega le risorse in modo esclusivo, con lo scopo di massimizzare la quantità di lavoro che l'utente può svolgere. In questo caso il sistema operativo si progetta considerando principalmente la **facilità d'uso**, con qualche attenzione alle prestazioni, ma nessuna all'**utilizzo delle risorse**, a come cioè sono condivise le risorse hardware e software. Le prestazioni sono naturalmente importanti; questi sistemi, tuttavia, si focalizzano sull'esperienza del singolo utente più che sull'utilizzo delle risorse in generale.

Alcuni utenti usano terminali connessi a **mainframe** o **minicalcolatori** condividendone le risorse con altri utenti anche loro connessi tramite terminali. In questo caso il sistema operativo si progetta per massimizzare l'utilizzo delle risorse, garantendo che tutto il tempo disponibile di CPU, la memoria e le periferiche di I/O siano impiegati in modo equo ed efficiente.

In altri casi ci sono utenti che usano **stazioni di lavoro**, connesse a reti di altre stazioni di lavoro e a **server**; dispongono di risorse loro riservate: altre devono condividerle, come la rete e i server, per l'accesso ai file e per i servizi di stampa e di elaborazione. Tutto ciò richie-

de che il sistema operativo sia progettato ponderando l'adeguatezza all'uso individuale e l'utilizzo ottimale delle risorse.

Negli ultimi tempi si sono diffusi calcolatori palmari dei tipi più svariati; si tratta prevalentemente di unità a sé stanti, usate singolarmente da ciascun utente. Alcuni sono collegati a reti di calcolatori, direttamente tramite cavi o più spesso tramite dispositivi d'accesso senza fili. A causa della loro scarsa autonomia e interfaccia limitata, svolgono un numero piuttosto ridotto di operazioni remote. I loro sistemi operativi sono progettati principalmente per facilitare l'uso individuale, ma anche per ridurre il consumo delle batterie.

Alcuni calcolatori sono integrati nei prodotti più vari (*embedded system*), e hanno poca o nessuna visibilità per gli utenti: i calcolatori integrati negli elettrodomestici e nelle automobili, per esempio, possono avere una tastiera numerica, e accendere o spegnere alcuni indicatori luminosi per segnalare il proprio stato; questi apparati e i relativi sistemi operativi, nella maggior parte dei casi, sono tuttavia progettati per funzionare senza l'intervento degli utenti.

1.1.2 Punto di vista del sistema

Dal punto di vista del calcolatore, il sistema operativo è il programma più strettamente correlato ai suoi elementi fisici. In tale contesto è possibile considerare un sistema operativo come un **assegnatore di risorse**. Un sistema di calcolo dispone di risorse (fisiche e programmi) utili per la risoluzione di un problema: tempo di CPU, spazio di memoria, spazio per la registrazione di file, dispositivi di I/O e così via. Il sistema operativo agisce come gestore di tali risorse. Di fronte a numerose ed eventualmente conflittuali richieste di risorse, il sistema operativo deve decidere come assegnarle agli specifici programmi e utenti affinché il sistema di calcolo operi in modo equo ed efficiente. Come abbiamo visto, l'assegnazione delle risorse è importante soprattutto nel caso in cui molti utenti accedono agli stessi mainfraim o minicalcolatori.

Una visione leggermente diversa di un sistema operativo enfatizza la necessità di controllare i dispositivi di I/O e i programmi utenti; un sistema operativo è in effetti un programma di controllo. Un **programma di controllo** gestisce l'esecuzione dei programmi utenti in modo da impedire che si verifichino errori o che il calcolatore sia usato in modo scorretto, soprattutto per quel che riguarda il funzionamento e il controllo dei dispositivi di I/O.

1.1.3 Definizione di sistema operativo

Abbiamo osservato il ruolo del sistema operativo sia dal punto di vista dell'utente sia da quello del sistema. Tuttavia, in generale, non si dispone di una definizione completa ed esauriente di sistema operativo. I sistemi operativi esistono poiché rappresentano una soluzione ragionevole al problema di realizzare un sistema di calcolo che si possa impiegare in modo utile, per eseguire i programmi utenti e facilitare la soluzione dei problemi degli utenti, ed è proprio per questo scopo che sono stati costruiti i calcolatori; ma, poiché un calcolatore di per sé non è molto facile da utilizzare, sono stati sviluppati i programmi applicativi. Questi programmi sono diversi tra loro, ma richiedono alcune funzioni comuni, per esempio il controllo dei dispositivi di I/O. Tali funzioni comuni, di controllo e assegnazione delle risorse, sono state racchiuse in un unico insieme coerente di programmi: il sistema operativo.

Non si dispone nemmeno di una definizione universalmente accettata di che cosa faccia parte di un sistema operativo. Un punto di vista semplice è considerare che esso comprenda tutto quello che il rivenditore fornisce quando gli si richiede "il sistema operativo". Tuttavia, i requisiti della memoria e della capacità di registrazione in dischi e nastri, e le funzioni richieste variano molto da sistema a sistema. Alcuni sistemi usano meno di un megabyte di memoria e

non possiedono neppure un *editor* a pieno schermo, mentre altri richiedono centinaia di megabyte di memoria e sono interamente basati su interfacce grafiche a finestre. La capacità di registrazione di un sistema si misura in gigabyte (un kilobyte o KB è pari a 1024 byte, un megabyte o MB è 1024^2 byte e un gigabyte o GB è 1024^3 byte; spesso i costruttori di calcolatori approssimano questi valori esprimendosi in termini di un milione di byte per un megabyte e di un miliardo di byte per un gigabyte). Una definizione più comune è quella secondo cui il sistema operativo è il solo programma che funziona sempre nel calcolatore, generalmente chiamato **kernel** (*nucleo*). (Oltre al kernel vi sono due tipi di programmi: i **programmi di sistema**, associati al sistema operativo, ma che non fanno parte del kernel, e i **programmi applicativi**, che includono tutti i programmi non correlati al funzionamento del sistema.)

La questione riguardante i componenti di un sistema operativo ha assunto una certa rilevanza anche in seguito all'azione legale promossa nel 1998 dal Dipartimento della giustizia degli Stati Uniti contro la Microsoft, accusata di includere troppe funzioni nel sistema operativo (come nel caso di programmi di consultazione del Web) e quindi di concorrenza sleale nei confronti dei produttori e rivenditori di applicazioni. Di conseguenza fu dichiarata colpevole di sfruttamento di monopolio a danno della concorrenza.

1.2 Organizzazione di un sistema di calcolo

Prima di addentrarci nello studio dei dettagli di un sistema operativo è necessaria una conoscenza generale della struttura di un calcolatore. Lo scopo del presente paragrafo è proprio consolidare le basi di questa conoscenza, proponendo una trattazione introduttiva dei diversi elementi di questa struttura. La trattazione riguarda principalmente le architetture dei calcolatori, quindi chi già conosce questo argomento può saltare il paragrafo o limitarsi a una semplice occhiata.

1.2.1 Funzionamento di un sistema di calcolo

Un moderno calcolatore d'uso generale è composto da una CPU e da un certo numero di controllori di dispositivi connessi attraverso un canale di comunicazione comune (*bus*) che permette l'accesso alla memoria condivisa dal sistema (Figura 1.2). Ciascuno di questi con-

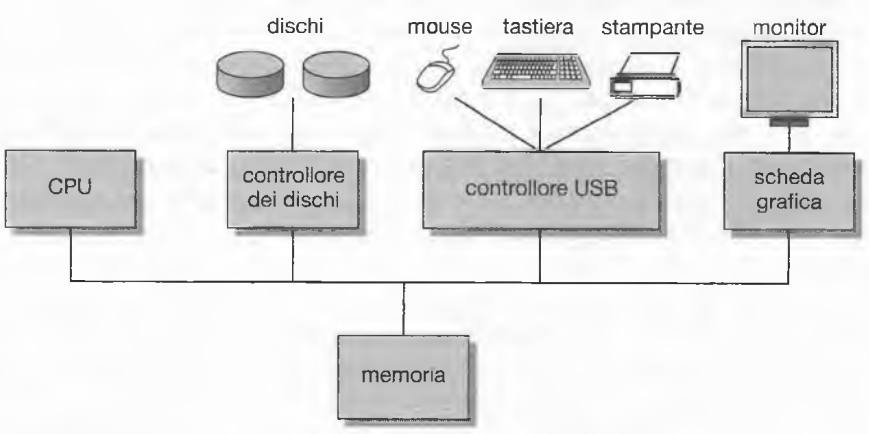

Figura 1.2 Moderno sistema di calcolo.

LO STUDIO DEI SISTEMI OPERATIVI

Lo studio dei sistemi operativi non è mai stato così interessante come al giorno d'oggi, e non è mai stato così facilitato. Con la diffusione del movimento open-source molti sistemi operativi, tra cui Linux, UNIX BSD, Solaris e una parte di Mac OS X sono diventati disponibili sia in formato sorgente sia in formato binario (eseguibile). Disponendo del codice sorgente è possibile studiare i sistemi operativi partendo dal loro interno e rispondere a domande che richiedevano in precedenza lo studio della documentazione o l'osservazione del comportamento di un sistema operativo.

Inoltre, l'incremento della virtualizzazione quale funzione principale (e spesso gratuita) del computer permette di far funzionare più sistemi operativi su un unico sistema (*core system*). Ad esempio, VMware (http://www.wmware.com) fornisce un programma gratuito sul quale possono girare centinaia di applicazioni virtuali gratuite. Grazie a questo metodo, gli studenti possono sperimentare senza alcun costo centinaia di sistemi operativi diversi all'interno di un sistema operativo dato.

Anche quei sistemi operativi non più attuali dal punto di vista commerciale sono spesso disponibili in versione open-source, il che permette di studiarne il funzionamento su sistemi con CPU lenta e poche risorse di memoria. All'indirizzo http://dmoz.org/Computers/Software/Operating_Systems/Open_Source/ è reperibile un'ampia lista, seppur non esaustiva, di progetti di sistemi operativi open-source. In alcuni casi sono disponibili anche simulatori di hardware specifico che consentono al sistema operativo di funzionare sull'hardware nativo, anche quando si stanno utilizzando un computer e un sistema operativo moderni. Ad esempio, un simulatore DECSYSTEM-20 attivo su Mac OS X può avviare TOPS-20, caricare nastri e modificare e compilare un nuovo kernel TOPS-20. Se interessato, lo studente può cercare in Internet i manuali e i documenti originali che descrivono il sistema operativo.

L'avvento dei sistemi operativi open-source accorcia la distanza tra studenti e sviluppatori di sistemi operativi. Con qualche conoscenza, un po' d'impegno e una connessione Internet, lo studente può persino creare una nuova distribuzione di un sistema operativo! Solo pochi anni fa era difficile, se non impossibile, accedere al codice sorgente, mentre al giorno d'oggi l'unica limitazione consiste nel tempo e nello spazio su disco di cui uno studente dispone.

trollori si occupa di un particolare tipo di dispositivo fisico (per esempio, unità a disco, dispositivi audio e unità video). La CPU e questi controllori possono operare in modo concorrente, contendendosi i cicli d'accesso alla memoria. La sincronizzazione degli accessi alla memoria è garantita dalla presenza di un controllore di memoria.

L'avviamento del sistema, conseguente all'accensione fisica di un calcolatore, così come il riavvio di un calcolatore già acceso, richiede la presenza di uno specifico programma iniziale, di solito non troppo complesso, detto **programma d'avviamento** (*bootstrap program*), in genere contenuto in tipi di memoria noti con il termine generale di *firmware*, il cui supporto fisico è parte integrante della macchina. Esempi di firmware sono le memorie a sola lettura (*read only memory*, ROM), e le memorie programmabili cancellabili elettricamente (EEPROM). La funzione di tale programma consiste nell'inizializzare i diversi componenti del sistema, dai registri della CPU ai controllori dei diversi dispositivi, fino al contenuto della memoria centrale. Il programma d'avviamento deve caricare nella memoria il sistema operativo e avviarne l'esecuzione, perciò individua e carica nella memoria il kernel del sistema operativo; il sistema operativo avvia quindi l'esecuzione del primo processo d'elaborazione, per esempio `init`, e attende che si verifichi qualche evento.

Un evento è di solito segnalato da un'interruzione dell'attuale sequenza d'esecuzione della CPU, che può essere causata da un dispositivo fisico o da un programma. Nel primo caso si parla di **segnale d'interruzione** o, più brevemente, **interruzione** (*interrupt*); si tratta di segnali che i controllori dei dispositivi e altri elementi dell'architettura possono inviare alla CPU, di solito attraverso il bus di sistema. Nel secondo caso si parla di **segnale di eccezione** o, più brevemente, **eccezione** (*exception* o *trap*), che può essere causata da un programma in esecuzione a seguito di un evento eccezionale, riconosciuto tramite l'architettura della CPU (per esempio un errore: una divisione per zero o un accesso alla memoria non valido); oppure a seguito di una richiesta specifica effettuata da un programma utente per ottenere l'esecuzione di un servizio del sistema operativo, attraverso una speciale istruzione detta **chiamata di sistema** (*system call*) o **chiamata supervisore** (*supervisor call*, SVC).

Ogniqualvolta riceve un segnale d'interruzione, la CPU interrompe l'elaborazione corrente e trasferisce immediatamente l'esecuzione a una locazione fissa della memoria. Di solito, questa locazione contiene l'indirizzo iniziale della procedura di servizio per quel dato segnale d'interruzione. Una volta completata l'esecuzione della procedura richiesta, la CPU riprende l'elaborazione precedentemente interrotta. La Figura 1.3 mostra il diagramma temporale della gestione di un simile evento.

I segnali d'interruzione sono un elemento importante nell'architettura di un calcolatore, e ciascun tipo di calcolatore ha il proprio meccanismo delle interruzioni; ciò nonostante molte funzioni sono comuni. Un segnale d'interruzione deve causare il trasferimento del controllo all'appropriata procedura di servizio dell'evento a esso associato. Il modo più semplice per gestire quest'operazione è quello di impiegare una procedura generale che esamina le informazioni presenti nel segnale d'interruzione, e invoca la procedura di gestione dello specifico segnale d'interruzione. D'altra parte, la gestione di un'interruzione deve essere molto rapida perciò, considerando che il numero dei possibili segnali d'interruzione è predefinito, si può usare una tabella di puntatori alle specifiche procedure. In questo modo, l'attivazione delle procedure di servizio delle interruzioni avviene in modo indiretto attraverso questa tabella, senza procedure intermedie. In genere la tabella di puntatori contenente gli indirizzi delle procedure di servizio delle interruzioni è mantenuta nella memoria bassa (per esempio, le prime 100 locazioni). L'accesso a questa sequenza d'indirizzi, detta **vettore delle interruzioni**, avviene per mezzo di un indice, codificato nello stesso segnale d'interruzione, allo scopo di fornire l'indirizzo della procedura di servizio relativa all'evento segnalato dall'interruzione. Sistemi operativi radicalmente differenti, come Windows e UNIX, usano lo stesso meccanismo di gestione delle interruzioni.

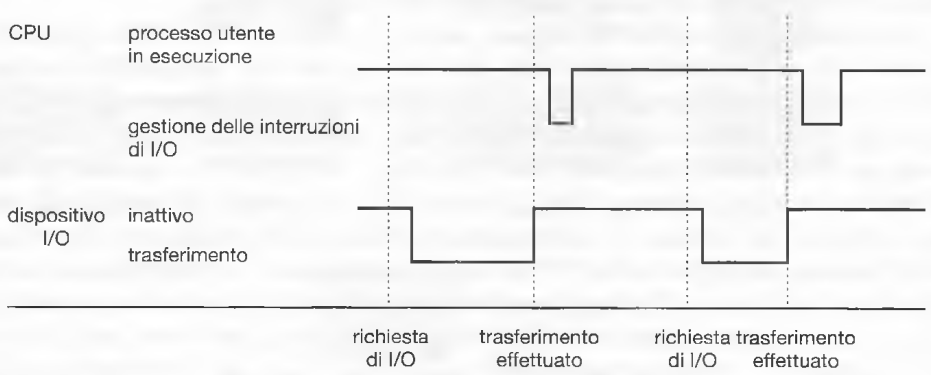

Figura 1.3 Diagramma temporale delle interruzioni per un singolo processo che emette dati.

L'architettura di gestione delle interruzioni deve anche salvare l'indirizzo dell'istruzione interrotta. Molti sistemi di vecchia concezione si limitavano a memorizzare l'indirizzo dell'istruzione interrotta in una locazione fissa o indicizzata dal numero di dispositivo; architetture più recenti memorizzano l'indirizzo di ritorno nella pila (*stack*) di sistema. Se la procedura di gestione dell'interruzione richiede la modifica dello stato della CPU, per esempio modificando il contenuto di qualche registro, deve salvare esplicitamente lo stato corrente per poterlo ripristinare prima di restituire il controllo. Terminato il servizio dell'interruzione, l'indirizzo di ritorno precedentemente salvato viene carica nel **contatore di programma** (*program counter*), che contiene l'indirizzo della prossima istruzione da eseguire, consentendo la ripresa della computazione interrotta come se nulla fosse accaduto.

1.2.2 Struttura della memoria

La CPU può caricare istruzioni esclusivamente dalla memoria, quindi tutti i programmi da eseguire devono esservi caricati. I computer general-purpose eseguono la maggior parte dei programmi da una memoria riscrivibile, la memoria principale, chiamata anche **memoria ad accesso diretto** (*random access memory*, RAM). La memoria principale è realizzata solitamente con una tecnologia basata su semiconduttori chiamata **memoria dinamica ad accesso diretto** (*dynamic random access memory*, DRAM). I computer utilizzano anche altri tipi di memoria. Dal momento che la memoria di sola lettura (ROM) non può essere modificata, solo i programmi statici vi sono salvati. L'immutabilità della ROM è utile nelle cartucce per i videogiochi. Le EEPROM non possono essere cambiate di frequente, e per questo contengono per lo più programmi statici. Sulle EEPROM degli smartphone, ad esempio, sono memorizzati i programmi installati inizialmente dalla fabbrica.

Tutte le tipologie di memoria forniscono un vettore di parole. Ciascuna parola possiede un proprio indirizzo. L'interazione avviene per mezzo di una sequenza di istruzioni `load` e `store` opportunamente indirizzate. L'istruzione `load` trasferisce il contenuto di una parola della memoria centrale in uno dei registri interni della CPU, mentre `store` copia il contenuto di uno di questi registri nella locazione di memoria specificata. Oltre agli accessi dovuti alle operazioni `load` e `store`, che si richiedono in modo esplicito, la CPU preleva automaticamente dalla memoria centrale le istruzioni da eseguire.

La tipica sequenza d'esecuzione di un'istruzione, in un sistema con architettura di **von Neumann**, comincia con il prelievo (*fetch*) di un'istruzione dalla memoria centrale e il suo trasferimento nel **registro d'istruzione**. Quindi si decodifica l'istruzione che eventualmente può richiedere il trasferimento di alcuni operandi dalla memoria in alcuni registri interni. Una volta terminata l'esecuzione dell'istruzione sugli operandi, il risultato si può scrivere nella memoria. Si noti che l'unità di memoria "vede" soltanto una sequenza d'indirizzi di memoria; non importa né il modo in cui questi sono stati generati (dal contatore di programma, per indicizzazione, riferimento indiretto, indirizzamento immediato, e così via) né tanto meno a che cosa fanno riferimento (istruzioni o dati). Di conseguenza, anche la presente trattazione non si cura di *come* questi indirizzi siano generati all'interno dei programmi e si occupa semplicemente delle sequenze d'indirizzi della memoria generate dai programmi in esecuzione.

In teoria, si vorrebbe che sia i programmi sia i dati da essi trattati potessero risiedere in modo permanente nella memoria centrale. Questo non è possibile per i seguenti due motivi:

1. la capacità della memoria centrale non è di solito sufficiente a contenere in modo permanente tutti i programmi e i dati richiesti;
2. la memoria centrale è un dispositivo di memorizzazione *volatile*, sicché perde il proprio contenuto quando si spegne il sistema o si ha un'interruzione dell'alimentazione elettrica.

Per queste ragioni la maggior parte dei sistemi di calcolo comprende una **memoria secondaria** come estensione della memoria centrale. La caratteristica fondamentale di questi dispositivi è la capacità di conservare in modo permanente grandi quantità di informazioni.

Il dispositivo più comunemente impiegato a questo scopo è l'unità a **disco magnetico**, adoperato per la memorizzazione sia di programmi sia di dati. La maggior parte dei programmi (elaboratori di testi, fogli di calcolo, programmi di consultazione del Web, compilatori, e così via) è mantenuta in un disco sino al momento del caricamento nella memoria, e fa uso di un disco come sorgente e destinazione delle informazioni elaborate. Come si spiega nel Capitolo 12, una corretta gestione delle unità a disco è di fondamentale importanza per un sistema di calcolo.

Occorre comunque dire che la struttura proposta (composta da registri, memoria centrale e unità a disco) rappresenta semplicemente una delle possibili configurazioni del sistema di memorizzazione di un calcolatore. Esistono altri tipi di memorie, per esempio le memorie cache, i CD-ROM e i nastri magnetici. Qualsiasi architettura fornisce le funzioni fondamentali che consentono la memorizzazione di un dato e il suo mantenimento fino all'uso successivo. Le caratteristiche che differenziano i diversi sistemi di memorizzazione sono velocità, costo, dimensioni e volatilità.

L'ampio ventaglio dei sistemi di memorizzazione disponibili in un elaboratore può essere ordinato secondo una scala gerarchica (Figura 1.4), sulla base della velocità e del costo. I gradini più alti ospitano i dispositivi più veloci, ma anche più dispendiosi. Andando verso il basso il costo per bit generalmente diminuisce, mentre il tempo di accesso tende ad aumentare. Si tratta di un compromesso ragionevole: se un certo sistema di memorizzazione, a parità di condizioni, fosse più veloce e nel contempo meno costoso rispetto ad altri, verrebbe meno qualunque motivo per usare la soluzione più lenta e costosa. In effetti, molti dispositivi di memorizzazione della prima ora, quali i sistemi a nastro perforato e le memorie a nuclei magnetici, sono ormai relegati nei musei, essendo stati soppiantati dai nastri magnetici e dalla me-

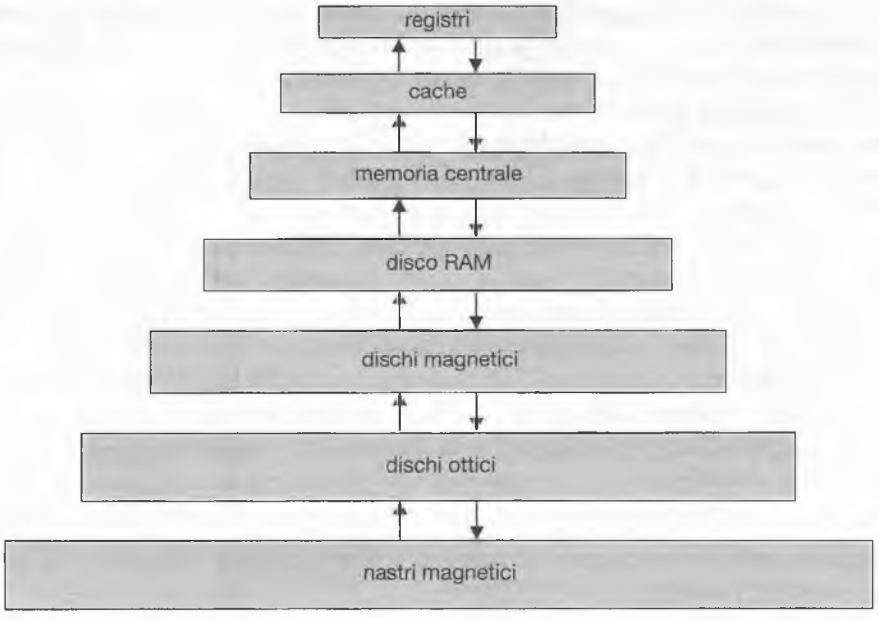

Figura 1.4 Gerarchia dei dispositivi di memoria.

moria a semiconduttori, più rapidi ed economici. I dispositivi nei quattro livelli superiori della Figura 1.4 possono essere materialmente costruiti con la memoria a semiconduttori.

Oltre a caratterizzarsi per velocità e costo, i sistemi di memorizzazione si suddividono in volatili e non volatili. Come si è accennato, la **memoria volatile** comporta la perdita dei dati nel caso di interruzione dell'alimentazione. Qualora non si disponga di sistemi di salvataggio automatico dei dati (*sistemi di backup*) basati su dispendiose batterie o generatori elettrici, affinché siano preservati è necessario salvare i medesimi su **memoria non volatile**. Nella gerarchia della Figura 1.4, i dispositivi di memorizzazione al di sopra del disco RAM sono volatili, mentre gli ultimi tre in basso sono non volatili. Un **disco** RAM si può progettare per essere volatile o non volatile. Durante il normale funzionamento, il disco RAM memorizza i dati in un capiente vettore DRAM, che è volatile. Tuttavia, molti dischi RAM contengono, in posizione nascosta, un disco rigido magnetico e un alimentatore di riserva per la batteria. Nel caso di un'interruzione della corrente elettrica, il controllore del disco RAM copia i dati dalla RAM sul disco magnetico, per poi eseguire l'operazione inversa al ripristino della corrente elettrica. Una variante del disco RAM è rappresentata dalla memoria flash, impiegata in vari prodotti, dalle macchine fotografiche agli **assistenti digitali personali** (PDA) e ai robot; la sua diffusione come memoria rimovibile nei computer di uso generale è in costante aumento. La memoria flash è più lenta della DRAM, ma non necessita di alimentazione esterna. Un'altra possibilità per la memorizzazione non volatile è la NVRAM, vale a dire la DRAM provvista di batterie di scorta. Questo tipo di memoria arriva a eguagliare la DRAM in velocità, ma la sua non volatilità ha durata limitata.

Nel progettare un sistema di memorizzazione completo si deve attribuire la giusta rilevanza a ciascun fattore: l'uso di memoria costosa va limitato al necessario; in compenso, è bene prevedere la massima quantità possibile di memoria non volatile ed economica. L'installazione di memorie cache sopperisce a eventuali macroscopiche disparità nei tempi di accesso o nella velocità di trasferimento tra due componenti, consentendo miglioramenti nelle prestazioni.

1.2.3 Struttura di I/O

La memoria è solo uno dei numerosi dispositivi di I/O di un elaboratore. Una percentuale cospicua del codice di un sistema operativo è dedicata alla gestione dell'I/O; ciò è dovuto in parte alla sua importanza per il progetto di un sistema affidabile ed efficiente e in parte alla natura variabile dei dispositivi preposti all'ingresso e all'uscita dei dati. Presentiamo adesso una panoramica generale dell'I/O.

Un calcolatore d'uso generale è composto da una CPU e da un insieme di controllori di dispositivi connessi mediante un bus comune. Ciascun controllore deve occuparsi di un particolare tipo di dispositivo e, secondo la sua natura, può gestire uno o più dispositivi a esso connessi. Un controllore SCSI (*small computer-systems interface*), per esempio, è capace di controllare sette o più dispositivi. Un controllore di dispositivo dispone di una propria memoria interna, detta memoria di transito (*buffer*), e di un insieme di registri specializzati. Il controllore è responsabile del trasferimento dei dati tra i dispositivi periferici a esso connessi e la propria memoria di transito. I sistemi operativi in genere possiedono, per ogni controllore del dispositivo, un **driver del dispositivo** che si coordina con il controllore e funge da interfaccia uniforme con il resto del sistema.

Per avviare un'operazione di I/O, il driver del dispositivo carica i registri interessati all'interno del controllore, il quale, dal canto suo, esamina i contenuti di questi registri per scegliere l'azione da intraprendere (per esempio "leggi un carattere dalla tastiera"). Il controllore comincia a trasferire i dati dal dispositivo al proprio buffer locale. A trasferimento

completato, il controllore informa il driver, tramite un'interruzione, di avere terminato l'operazione. Il driver passa quindi il controllo al sistema operativo, restituendo i dati (o un puntatore a essi) se l'operazione è di lettura; per altre operazioni, il driver restituisce delle informazioni di stato.

Questa forma di I/O guidato dalle interruzioni è adatto al trasferimento di piccole quantità di dati, ma in caso di trasferimenti massicci può generare un pesante sovraccarico; si pensi, per esempio, all' I/O da e verso il disco. Per fare fronte a questo problema si utilizza le tecnica dell'**accesso diretto alla memoria** (DMA). Una volta impostati i buffer, i puntatori e i contatori necessari al dispositivo di I/O, il controllore trasferisce un intero blocco di dati dalla propria memoria buffer direttamente nella memoria centrale, o viceversa, senza alcun intervento da parte della CPU. In questo modo l'operazione richiede una sola interruzione per ogni blocco di dati trasferito, piuttosto che per ogni byte, come avviene nel caso dei dispositivi più lenti. Così, mentre il controllore del dispositivo effettua le operazioni descritte, la CPU rimane libera e può occuparsi di altri compiti.

Alcuni sistemi all'avanguardia hanno abbandonato la configurazione basata sul bus per adottare un'architettura incentrata sugli switch, in cui più dispositivi fisici possono interagire con varie parti del sistema concorrentemente, piuttosto che contendersi un unico bus condiviso. In tali circostanze, l'accesso diretto alla memoria risulta ancora più efficace. La Figura 1.5 mostra l'interazione di tutti i componenti di un elaboratore.

1.3 Architettura degli elaboratori

Nel Paragrafo 1.2 è stata presentata la struttura generale di un calcolatore, che può essere organizzato secondo criteri molto diversi; in prima battuta faremo riferimento al numero di unità di elaborazione presenti in elaboratori d'uso generale.

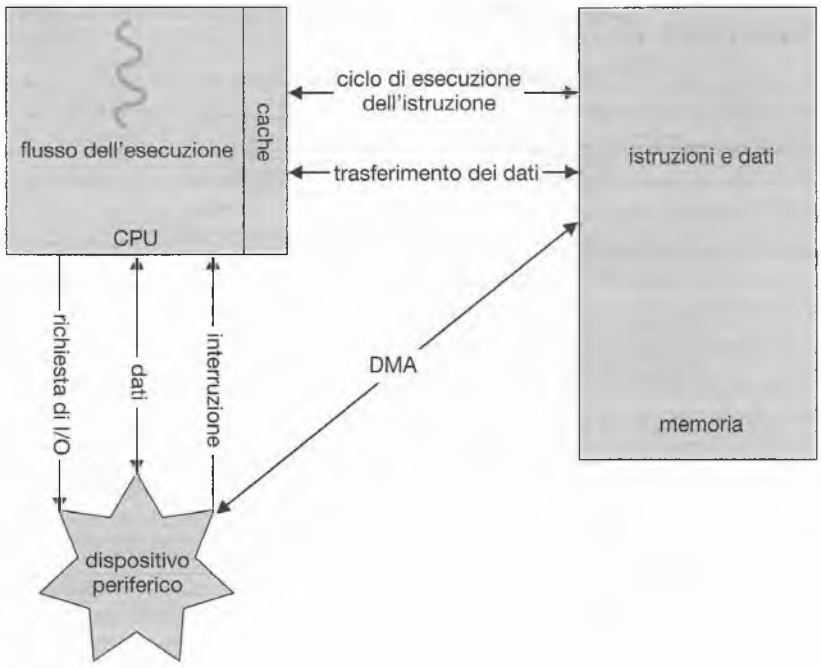

Figura 1.5 Funzionamento di un moderno sistema operativo.

1.3.1 Sistemi monoprocessore

Sono molti i sistemi che usano un solo processore. La loro varietà può risultare stupefacente: si va dai PDA fino ai mainframe. Un sistema monoprocessore è dotato di una CPU principale in grado di eseguire un insieme di istruzioni di natura generale, comprese quelle necessarie ai processi utenti. Quasi tutti i sistemi, inoltre, possiedono altri processori specializzati, deputati a compiti particolari. Essi possono assumere la forma di processori specifici di un dispositivo, quali i controllori del disco, della tastiera o del video; oppure, quando risiedono su mainframe, essere impiegati per finalità più generiche, come i processori dell'I/O che spostano dati da un dispositivo a un altro del sistema.

Tutti questi processori di tipo specifico sono dotati di un insieme ristretto di istruzioni, e non eseguono processi utenti. Talvolta sono guidati dal sistema operativo, che può inviare loro informazioni sul compito da espletare successivamente, e controllarne lo status. Prendiamo l'esempio di un microprocessore che funge da controllore del disco e che riceve dalla CPU un elenco di richieste; spetta a esso implementare una coda e un algoritmo di scheduling per gestirle. Questa strategia alleggerisce la CPU dal sovraccarico di lavoro che lo scheduling del disco può comportare. I PC ospitano un microprocessore all'interno della tastiera, per convertire la pressione di ciascun tasto nel codice appropriato da trasmettere alla CPU. In circostanze o sistemi differenti, i processori con finalità specifiche sono dispositivi di basso livello integrati nell'hardware. Il sistema operativo non può comunicare con questi processori, che svolgono in autonomia il proprio lavoro. L'utilizzo di microprocessori con finalità specifiche è comune e non trasforma un sistema monoprocessore in un sistema multiprocessore. Se è presente una sola CPU, si tratta di un sistema monoprocessore.

1.3.2 Sistemi multiprocessore

Sebbene i sistemi monoprocessore siano ancora molto comuni, l'importanza dei sistemi multiprocessore, conosciuti anche come sistemi paralleli o sistemi strettamente connessi (*tightly coupled system*), è in rapida crescita. Questo tipo di sistemi dispone di più unità d'elaborazione in stretta comunicazione, che condividono i canali di comunicazione all'interno del calcolatore (*bus*), i timer dei cicli di macchina (*clock*) e talvolta i dispositivi di memorizzazione e periferici.

Tali sistemi hanno tre vantaggi principali.

1. **Maggiore produttività** (*throughput*). Aumentando il numero di unità d'elaborazione è possibile svolgere un lavoro maggiore in meno tempo. Con n unità d'elaborazione la velocità non aumenta tuttavia di n volte, ma in misura minore. Infatti, se più unità d'elaborazione collaborano nell'esecuzione di un compito, il sistema operativo deve gestire le operazioni che garantiscono che tutti i componenti funzionino correttamente. Questo sovraccarico, unito alla contesa delle risorse condivise, riduce il guadagno atteso dalla disponibilità di più unità d'elaborazione; così come un gruppo di n programmatori che lavorano insieme non produce n volte quanto produrrebbe un solo programmatore.
2. **Economia di scala.** I sistemi multiprocessore possono inoltre consentire risparmi rispetto a più sistemi dotati di una sola unità d'elaborazione, poiché nei primi si possono condividere dispositivi periferici, mezzi di registrazione dei dati e alimentatori elettrici. Se più programmi devono operare sullo stesso insieme di dati, è economicamente più conveniente registrarli in dischi condivisi da tutte le unità d'elaborazione, piuttosto che avere più calcolatori con i rispettivi dischi locali e più copie degli stessi dati.

3. **Incremento dell'affidabilità.** Se le funzioni si possono distribuire adeguatamente tra più unità d'elaborazione, un guasto di alcune di loro non blocca il sistema, semplicemente lo rallenta; ciascuna delle unità d'elaborazione rimanenti assume su di sé una parte del lavoro che svolgevano le unità d'elaborazione guaste; l'intero sistema non si ferma, ma funziona a una velocità ridotta.

Per molte applicazioni una maggiore affidabilità dell'elaboratore è di importanza cruciale. La capacità di continuare a offrire un servizio proporzionalmente commisurato, per qualità, ai dispositivi ancora in funzione è detta **degradazione controllata** (*graceful degradation*). Taluni sistemi si spingono oltre la degradazione controllata e sono così definiti **tolleranti ai guasti** (*fault-tolerant*) perché, nonostante subiscano il danneggiamento di un qualunque singolo componente, continuano ugualmente a funzionare. La resistenza ai guasti, si noti, necessita di un meccanismo per il riconoscimento del danno, la sua diagnosi e, se è possibile, la riparazione. Il sistema HP NonStop (già noto come Tandem) duplica sia i dispositivi sia i programmi per garantire la continuità di funzionamento anche in caso di guasti. Esso è formato da coppie multiple della CPU, la cui attività è sincronizzata. Entrambi i processori di una coppia eseguono una data istruzione e confrontano i risultati. Risultati diversi segnalano un errore da parte di una delle CPU, e provocano il blocco di entrambe. Il processo che era in esecuzione è trasferito quindi a un'altra coppia di CPU, e riprende l'esecuzione a partire dall'istruzione a cui era avvenuto il blocco. Si tratta di una soluzione costosa, dato che implica l'uso di dispositivi dedicati e una notevole duplicazione delle risorse fisiche.

I sistemi multiprocessore attualmente in uso sono di due tipi. Alcuni impiegano la **multielaborazione asimmetrica** (*asymmetric multiprocessing*, AMP), in cui a ogni unità d'elaborazione si assegna un compito specifico. Un'unità d'elaborazione principale controlla il sistema, le altre attendono istruzioni dall'unità principale oppure hanno compiti predefiniti. Questo schema definisce una relazione gerarchica; l'unità d'elaborazione principale organizza e assegna il lavoro alle unità d'elaborazione secondarie.

Nei sistemi più comuni si ricorre alla **multielaborazione simmetrica** (*symmetric multiprocessing*, SMP), in cui ogni processore è abilitato al compimento di tutte le operazioni del sistema. La tecnica SMP pone tutti i processori su un piano di parità; tra essi non vi è subordinazione gerarchica. La Figura 1.6 illustra una tipica architettura SMP. Un esempio della tecnica SMP è dato da Solaris, realizzato da Sun Microsystems quale versione commerciale di UNIX. Tale sistema può impiegare dozzine di processori, tutti gestiti dal sistema operativo Solaris. Il vantaggio offerto da questo modello è che molti processi sono eseguibili contemporaneamente (*n* processi se si hanno *n* CPU) senza causare un rilevante calo delle prestazioni. Per essere certi che i dati raggiungano le unità d'elaborazione giuste occorre però con-

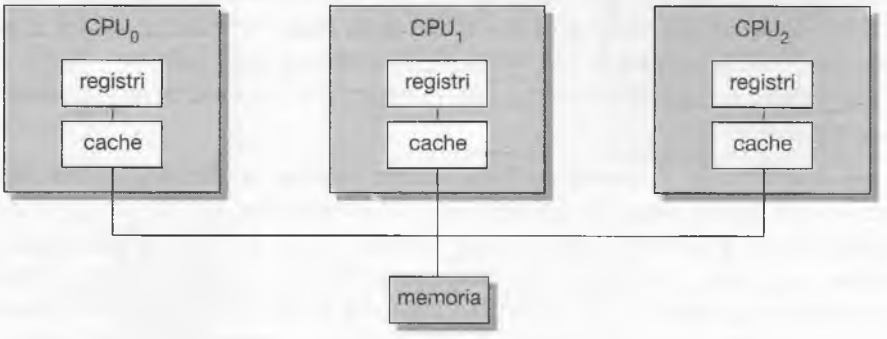

Figura 1.6 Architettura per la multielaborazione simmetrica.

trollare con molta attenzione le operazioni di I/O. Inoltre, poiché le unità d'elaborazione sono separate, una potrebbe essere inattiva mentre un'altra è sovraccarica, e ciò determinerebbe un'inefficienza evitabile se le unità d'elaborazione condividessero alcune strutture dati. Un sistema multiprocessore di questo tipo permetterebbe infatti di condividere dinamicamente processi e risorse (per esempio la memoria) tra le varie unità d'elaborazione, riducendo la varianza tra di loro. Come si vedrà nel Capitolo 6, un sistema di questo tipo deve essere scritto con molta cura. Tutti i sistemi operativi moderni (tra i quali il Windows, Windows XP, Mac OS X e Linux) ora gestiscono la multielaborazione simmetrica.

La differenza tra multielaborazione simmetrica e asimmetrica può derivare da caratteristiche sia dell'architettura del sistema sia da caratteristiche del sistema operativo: una speciale architettura può differenziare le unità d'elaborazione, oppure si può avere un sistema operativo che definisce una sola unità d'elaborazione primaria e più unità d'elaborazione secondarie. Per esempio, la versione 4 del sistema operativo SunOS della Sun consentiva la multielaborazione asimmetrica, mentre la versione 5 (Solaris) permette, sulla stessa architettura, la multielaborazione simmetrica.

Per aumentare la potenza di calcolo nella multielaborazione si aggiungono nuove CPU. Se la CPU ha un controllo di memoria integrato, allora l'aggiunta di CPU può aumentare la quantità di memoria indirizzabile dal sistema. D'altra parte, la multielaborazione può causare il cambiamento del modello di accesso alla memoria del sistema, trasformando un accesso uniforme alla memoria (UMA) in accesso non uniforme alla memoria (NUMA). Per accesso uniforme alla memoria si intende la situazione in cui l'accesso a una RAM da una qualsiasi CPU richiede lo stesso tempo. In caso di NUMA, invece, l'accesso ad alcune parti della memoria necessita di un tempo maggiore rispetto ad altre, con un conseguente peggioramento delle prestazioni. I sistemi operativi possono minimizzare la penalizzazione causata dal NUMA grazie a un'oculata gestione delle risorse, come discusso nel Paragrafo 9.5.4.

Una tendenza recente nella progettazione della CPU è raggruppare diverse unità di calcolo (*core*) in un singolo circuito. Si tratta, sostanzialmente, di circuiti integrati multiprocessore. Questi circuiti possono essere più efficienti rispetto a più circuiti dotati di una singola unità di calcolo, perché la comunicazione all'interno di un singolo circuito è più veloce rispetto a quella tra un circuito e un altro. Inoltre, un circuito dotato di diverse unità di calcolo usa molta meno potenza di diversi circuiti con una singola unità di calcolo. Ne consegue che i sistemi a multipla unità di calcolo sono particolarmente indicati per sistemi server come banche dati e server web.

Nella Figura 1.7 è illustrata un'architettura a doppia unità di calcolo (*dual core*), con due unità sullo stesso circuito. In un'architettura di questo tipo ogni unità di calcolo ha il proprio insieme di registri e la propria cache; altre architetture possono prevedere l'utilizzo di una cache condivisa oppure una combinazione di cache locali e condivise. A prescindere dalle considerazioni riguardanti le architetture, come la competizione per l'uso della cache, della memoria e del bus, queste CPU a unità multipla appaiono al sistema operativo come *N* processori ordinari. Chi progetta sistemi operativi e chi programma applicazioni è quindi spinto all'impiego di questo tipo di CPU.

Infine, accenniamo a una delle ultime innovazioni, i cosiddetti **server blade**, che accolgono nello stesso contenitore fisico le schede del processore, dell'I/O e della rete. A differenza dei tradizionali sistemi multiprocessore, nei server blade ogni scheda madre (una scheda, cioè, che ospita una CPU) avvia ed esegue in maniera indipendente il proprio sistema operativo. Alcune di queste schede, poi, possono essere a loro volta multiprocessore, il che rende più labile la distinzione tra tipi diversi di computer. Si può affermare, in sintesi, che tali server sono costituiti da svariati sistemi multiprocessore indipendenti.

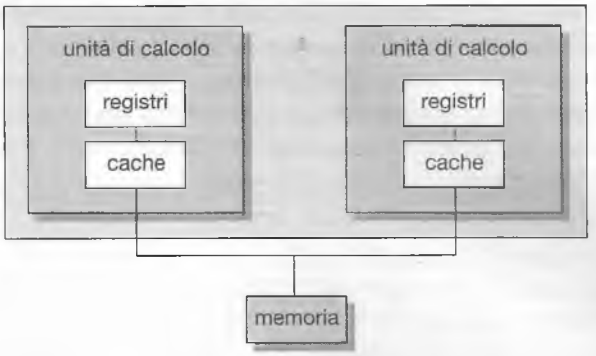

Figura 1.7 Architettura a doppia unità di calcolo, con due unità sullo stesso circuito.

1.3.3 Cluster di elaboratori

Come per i sistemi multiprocessore, i cluster di elaboratori (*clustered systems*) o cluster sono basati sull'uso congiunto di più unità d'elaborazione riunite per lo svolgimento di attività d'elaborazione comuni. Differiscono dai sistemi paralleli per il fatto che sono composti di due o più calcolatori completi collegati tra loro. In realtà per tali sistemi non esiste una definizione precisa; c'è un dibattito aperto tra i fornitori delle varie offerte commerciali su tale definizione e sul perché una soluzione sia migliore di un'altra. La definizione generalmente accettata è che si tratta di calcolatori che condividono la memoria di massa, connessi per mezzo di una rete locale (LAN), (Paragrafo 1.10), o connessioni più veloci come per esempio InfiniBand.

Di solito si adotta questo tipo di soluzione per offrire un'**elevata disponibilità**. Ciascun calcolatore esegue una serie di programmi che forma uno strato di gestione del cluster; ogni nodo può tenere sotto controllo (attraverso la LAN) uno o più degli altri nodi. Se si presenta un malfunzionamento, il calcolatore che svolge il controllo può appropriarsi dei mezzi di memorizzazione del calcolatore malfunzionante e riavviare le applicazioni che erano in esecuzione. Gli utenti e i client delle applicazioni notano solo una breve interruzione del servizio.

I cluster di elaboratori sono strutturabili in modo sia asimmetrico sia simmetrico. Nei **cluster asimmetrici** un calcolatore rimane nello stato di **attesa attiva** (*hot standby mode*) mentre l'altro esegue le applicazioni. Il primo non fa altro che tenere sotto controllo il server attivo (il secondo calcolatore). Se questo presenta un problema, il calcolatore di controllo diventa il server attivo. Nei **cluster simmetrici** due o più calcolatori eseguono le applicazioni e allo stesso tempo si controllano reciprocamente; in questo modo si ottiene una maggiore efficienza, poiché si utilizzano meglio le risorse, ma si richiede che siano disponibili più applicazioni da eseguire.

I cluster, essendo formati da diversi sistemi di computer collegati in rete, possono anche essere utilizzati per ottenere ambienti di elaborazione ad alte prestazioni (*high-performance computing*). Sistemi di questo tipo offrono molta più potenza di calcolo rispetto a un monoprocessore o persino rispetto a sistemi SMP, perché permettono l'esecuzione contemporanea di un'applicazione su tutti i computer del cluster. Tuttavia, le applicazioni devono essere scritte specificatamente in modo da trarre vantaggio dal cluster sfruttando una tecnica chiamata **parallelizzazione** (*parallelization*). Essa consiste nel suddividere il programma in componenti separate, eseguibili in parallelo su singoli computer all'interno del cluster. In genere tali applicazioni sono progettate in modo tale che, una volta che ogni nodo di elaborazione del cluster abbia risolto la sua porzione di problema, tutti i risultati vengano combinati in un'unica soluzione finale.

CLUSTER "BEOWULF"

I cluster di tipo Beowulf sono progettati per risolvere problemi di calcolo che richiedono elevate prestazioni. Sono costruiti utilizzando prodotti hardware, ad esempio dei personal computer, collegati da una semplice rete locale. La peculiarità più interessante dei cluster Beowulf consiste nel fatto che non utilizzano degli specifici pacchetti software, ma piuttosto una serie di librerie di software open-source che permettono ai nodi di elaborazione del cluster di comunicare. Esiste quindi una grande varietà di approcci per la costruzione di un cluster Beowulf, sebbene i nodi di elaborazione montino solitamente il sistema operativo Linux. I cluster Beowulf, non richiedendo hardware specifico e impiegando software open-source disponibile gratuitamente, offrono una soluzione a basso costo per la costruzione di un cluster ad alte prestazioni. Alcuni cluster Beowulf costituiti da vecchi personal computer sono formati da centinaia di nodi di elaborazione e utilizzati nel calcolo scientifico per risolvere problemi molto impegnativi dal punto di vista computazionale.

Altre forme sono i cluster di sistemi paralleli e quelli di sistemi connessi attraverso reti geografiche (WAN). I primi permettono a più calcolatori di accedere agli stessi dati nella memoria di massa condivisa. Poiché la maggior parte dei sistemi operativi non consente quest'accesso simultaneo ai dati da parte di più calcolatori, si ricorre a programmi specifici e particolari versioni delle applicazioni. Per esempio, l'Oracle Real Application Cluster è una versione del sistema di gestione delle basi di dati Oracle, progettata per funzionare in cluster di sistemi paralleli. Ogni calcolatore esegue l'applicazione Oracle e uno strato di programmi controlla l'accesso ai dischi condivisi, in questo modo ogni calcolatore del sistema ha accesso alla base di dati. Per ottenere questo accesso condiviso ai dati, il sistema deve anche prevedere il controllo dell'accesso e la mutua esclusione, in modo da evitare sul nascere i conflitti tra operazioni. Questa funzione, detta **gestione distribuita degli accessi** (*distributed lock manager*, DLM), è attiva in alcuni cluster di elaboratori.

La ricerca in quest'ambito sta attraversando una fase di rapida evoluzione. Alcuni prodotti di questo genere ospitano dozzine di sistemi in un solo cluster, e possono accorpare in un unico insieme nodi distanti chilometri. Tali progressi si devono in buona misura a **reti di memoria secondaria** (*storage-area network*, SAN), illustrati nel Paragrafo 12.3.3, grazie alle quali è possibile far sì che molti sistemi accedano a un'unica riserva di spazio per archiviare informazioni. Se le applicazioni e i relativi dati sono memorizzati in una SAN, il software del cluster può smistare l'applicazione verso una qualunque delle macchine che vi hanno accesso. Qualora venga meno una di loro, può subentrare qualsiasi altra macchina. Se una base di dati è implementata su un cluster di sistemi, decine di macchine possono condividerne il contenuto, con notevole aumento delle prestazioni e dell'affidabilità. La Figura 1.8 rappresenta la struttura generale di un cluster.

1.4 Struttura del sistema operativo

Avendo passato in rassegna la struttura e l'organizzazione dei sistemi informatici, siamo pronti ad affrontare i sistemi operativi. Il sistema operativo costituisce l'ambiente esecutivo dei programmi. Sebbene la struttura dei sistemi operativi possa variare grandemente, vi sono alcuni aspetti comuni che esponiamo in questo paragrafo.

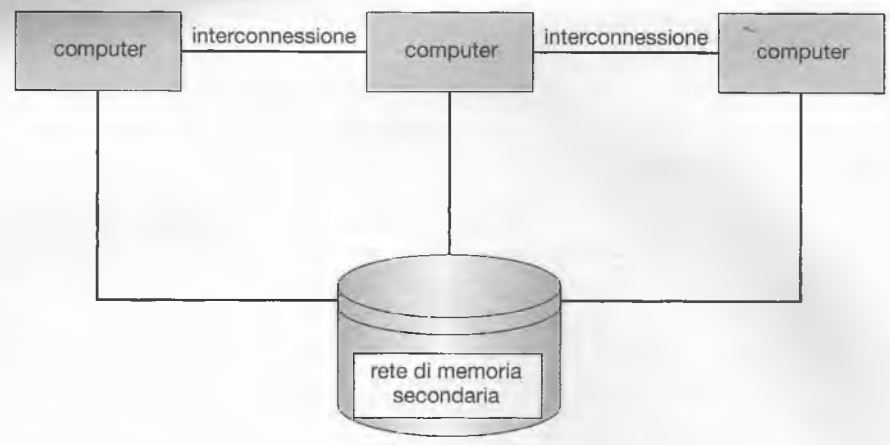

Figura 1.8 Struttura generale di un cluster.

Fra le più importanti caratteristiche dei sistemi operativi vi è la multiprogrammazione. In generale, un singolo utente non è in grado di tenere costantemente occupati la CPU e i dispositivi di I/O: la **multiprogrammazione** consente di aumentare la percentuale d'utilizzo della CPU, organizzando i lavori in modo tale da mantenerla in continua attività.

L'idea su cui si fonda questa tecnica è la seguente: il sistema operativo tiene contemporaneamente in memoria centrale diversi lavori (Figura 1.9). Dato che, in genere, la memoria centrale è troppo piccola per contenere tutti i programmi da eseguire, questi vengono collocati inizialmente sul disco in un'area apposita, detta **job pool**, contenente tutti i processi in attesa di essere allocati nella memoria centrale.

L'insieme dei programmi caricati in memoria è generalmente un sottoinsieme dei lavori contenuti nel *job pool*. Il sistema operativo ne sceglie uno tra quelli contenuti nella memoria e inizia l'esecuzione: a un certo punto potrebbe trovarsi nell'attesa di qualche evento, come il completamento di un'operazione di I/O. In questi casi, in un sistema non multiprogrammato, la CPU rimarrebbe inattiva. In un sistema con multiprogrammazione, invece, il sistema operativo passa semplicemente a un altro lavoro e lo esegue. Quando il primo lavoro ha terminato l'attesa, la CPU ne riprende l'esecuzione. Finché c'è almeno un lavoro da eseguire, la CPU non è mai inattiva.

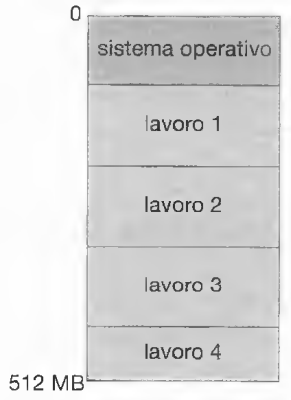

Figura 1.9 Configurazione della memoria per un sistema con multiprogrammazione.

Ritroviamo quest'idea anche in altre circostanze della vita comune. Un avvocato, per esempio, non lavora per un solo cliente alla volta: mentre un caso aspetta di essere dibattuto o si attende la stesura dei relativi documenti, l'avvocato può lavorare a un altro caso; se ha abbastanza clienti, non sarà mai inattivo per mancanza di lavoro. (Gli avvocati inattivi tendono a trasformarsi in politici, quindi tenerli occupati ha un certo valore sociale.)

Un sistema con multiprogrammazione fornisce un ambiente in cui le risorse del sistema (per esempio CPU, memoria, dispositivi periferici) sono impiegate in modo efficiente, ma non rappresenta un sistema d'interazione con l'utente. La **partizione del tempo d'elaborazione** (*time sharing* o *multitasking*) è un'estensione logica della multiprogrammazione; la CPU esegue più lavori commutando le loro esecuzioni con una frequenza tale da permettere a ciascun utente l'interazione col proprio programma durante la sua esecuzione.

Un **sistema di calcolo interattivo** permette la comunicazione diretta tra utente e sistema. L'utente impartisce le istruzioni direttamente al sistema operativo oppure a un programma, attraverso una tastiera o un mouse, e attende una risposta immediata. Il **tempo di risposta** dovrebbe perciò essere breve, in genere meno di un secondo.

Un sistema operativo a partizione del tempo d'elaborazione permette a più utenti di condividere contemporaneamente il calcolatore. Poiché le azioni e i comandi eseguiti in un sistema a partizione del tempo sono tendenzialmente brevi, a ciascun utente basta poter usare una piccola parte del tempo di calcolo della CPU. Il sistema passa rapidamente da un utente all'altro, quindi ogni utente ha l'impressione di disporre dell'intero calcolatore, che in realtà è condiviso da molti utenti.

Per assicurare a ciascun utente una piccola frazione del tempo di calcolo, un sistema operativo a partizione del tempo d'elaborazione si avvale dello scheduling della CPU e della multiprogrammazione. Ciascun utente dispone di almeno un proprio programma in memoria. Un programma caricato in memoria e predisposto per la fase d'esecuzione è noto come **processo**. Normalmente un processo, durante la sua esecuzione, impegna la CPU per un breve periodo di tempo prima di richiedere operazioni di I/O o di terminare; tali operazioni possono essere interattive, cioè i risultati sono inviati a uno schermo a disposizione dell'utente, che immette dati tramite una tastiera, un mouse, o altro. I tempi di tali operazioni risentono della lentezza del lavoro umano; l'immissione di dati e comandi tramite una tastiera, per esempio, è limitata dalla velocità di battitura dell'utente, che, per elevata che sia, è sempre molto bassa rispetto ai tempi d'elaborazione di un calcolatore: sette caratteri al secondo sono tanti per una persona, ma pochissimi per un calcolatore. Anziché lasciare inattiva la CPU, durante l'immissione interattiva il sistema operativo commuta rapidamente la CPU al programma di un altro utente.

La partizione del tempo di elaborazione e la multiprogrammazione richiedono la contemporanea presenza di diversi processi in memoria. Se alcuni processi sono pronti per il trasferimento in memoria centrale, ma lo spazio disponibile non è sufficiente per accoglierli tutti, il sistema deve fare una selezione; questa scelta, illustrata nel Capitolo 5, si chiama **job scheduling**, ossia *pianificazione dei lavori*. Quando il sistema operativo seleziona un processo dall'insieme dei lavori, carica quel processo in memoria perché sia eseguito. La coesistenza di un certo numero di programmi in memoria nello stesso lasso di tempo richiede una qualche forma di gestione della memoria, argomento esposto nei Capitoli 8 e 9. Inoltre, l'esistenza di diversi processi pronti per l'esecuzione nello stesso istante impone al sistema di scegliere solo alcuni di essi: i problemi inerenti a questa scelta, ovvero lo **scheduling della CPU** (*pianificazione della CPU*), sono descritti nel Capitolo 5. Infine, l'esecuzione concorrente di processi multipli rende necessario limitare il più possibile le loro interferenze reciproche sotto ogni aspetto del funzionamento del sistema, compresi lo scheduling dei pro-

cessi e la gestione del disco e della memoria. Tali problematiche sono affrontate di volta in volta attraverso il libro.

In un sistema basato sulla partizione del tempo di elaborazione, il sistema operativo deve garantire tempi di risposta accettabili: questa finalità è raggiunta, in alcuni casi, grazie alla tecnica detta **swapping** (*avvicendamento*), che consente di scambiare i processi presenti in memoria con quelli che risiedono su disco e viceversa. Un metodo più comune per ottenere il medesimo risultato è la **memoria virtuale**, tecnica che consente l'esecuzione di lavori d'elaborazione anche non interamente caricati nella memoria (Capitolo 9). Il più evidente vantaggio della memoria virtuale è che i programmi possono avere dimensioni maggiori della **memoria fisica**; inoltre, essa astrae la memoria centrale in un grande e uniforme vettore, separando la **memoria logica**, vista dall'utente, dalla memoria fisica, sollevando i programmatori dai problemi legati ai limiti della memoria.

I sistemi a partizione del tempo devono inoltre fornire un *file system* (Capitoli 10 e 11) residente in un insieme di dischi, i quali a loro volta necessitano di una gestione (Capitolo 12). I sistemi a partizione del tempo dispongono di meccanismi per la protezione delle risorse rispetto a utenti non autorizzati (Capitolo 14). Per assicurare che tale esecuzione sia disciplinata il sistema deve fornire meccanismi per la comunicazione e sincronizzazione dei processi (Capitolo 6) e garantire che i processi non s'inceppino in una situazione di stallo, in un'indefinita attesa reciproca (Capitolo 7).

1.5 Attività del sistema operativo

Come si è avuto modo di accennare, i moderni sistemi operativi sono guidati dalle interruzioni. Quando i dispositivi dell'I/O non richiedono alcun servizio, in assenza di processi da eseguire e di utenti a cui rispondere, il sistema operativo rimane inerte e attende che accada qualcosa. Quasi sempre, è un'interruzione o un'eccezione a segnalare gli eventi. Un segnale di eccezione (*trap* o *exception*) indica che si è verificata un'interruzione generata da un programma, dovuta a un errore (per esempio, una divisione per zero o l'accesso illegale alla memoria), o alla richiesta di erogazione, da parte di un programma utente, di uno dei servizi del sistema operativo. La struttura generale di un sistema operativo è definita proprio in quanto esso è guidato dalle interruzioni. A ciascun tipo di interruzione corrispondono nel sistema singoli segmenti di codice, che determinano la reazione all'interruzione; apposite routine per il servizio delle interruzioni hanno il compito di fornire loro una risposta adeguata.

Dal momento che il sistema operativo e gli utenti condividono la dotazione di programmi e dispositivi dell'elaboratore, è necessario assicurarsi che un errore provocato da un programma utente non possa arrecare danno ad altri programmi. In un ambiente condiviso, infatti, l'errore di un solo programma ha un effetto negativo potenziale sugli altri processi. Se, per esempio, un processo rimane bloccato in un ciclo infinito, potrebbe essere impedito il corretto funzionamento di molti altri processi. I sistemi multiprogrammati, inoltre, sono esposti a rischi più sottili; si pensi a un programma viziato da errori, capace di modificare interamente un altro programma, i suoi dati e persino lo stesso sistema operativo.

Senza un'efficace protezione da errori come quelli menzionati, il calcolatore è costretto a eseguire un solo processo per volta, a meno di non voler considerare sospetti tutti i dati in uscita. Un sistema operativo progettato in maniera soddisfacente deve evitare che un programma viziato da errori (oppure volutamente dannoso) possa indurre un'esecuzione scorretta di altri programmi.

1.5.1 Duplice modalità di funzionamento

Per garantire il corretto funzionamento del sistema è necessario distinguere tra l'esecuzione di codice del sistema operativo e di codice definito dall'utente. Il metodo seguito da molti sistemi operativi consiste nel disporre di specifiche caratteristiche dell'architettura del sistema che consentono di gestire differenti modalità di funzionamento.

Sono necessarie almeno due diverse modalità: **modalità utente** e **modalità di sistema** (detta anche *modalità kernel, modalità supervisore, modalità monitor* o *modalità privilegiata*). Per indicare quale sia la modalità attivo, l'architettura della CPU deve essere dotata di un bit, chiamato appunto **bit di modalità**: di sistema (0) o utente (1). Questo bit consente di stabilire se l'istruzione corrente si esegue per conto del sistema operativo o per conto di un utente. Quando l'elaboratore agisce per conto di un'applicazione utente, il sistema è in modalità utente. Tuttavia, allorquando l'applicazione utente rivolga una richiesta di servizio al sistema operativo (tramite una chiamata di sistema), per soddisfare la richiesta questi deve passare dalla modalità utente alla modalità di sistema. Ciò è esemplificato dalla Figura 1.10. Come vedremo, questa miglioria architetturale agevola il funzionamento del sistema anche in altre circostanze.

All'avviamento del sistema, il bit è posto in modalità di sistema. Si carica il sistema operativo che provvede all'esecuzione dei processi utenti in modalità utente. Ogni volta che si verifica un'interruzione o un'eccezione si passa dalla modalità utente a quella di sistema, cioè si pone a 0 il bit di modo. Perciò quando il sistema operativo riprende il controllo del calcolatore si trova in modalità di sistema. Prima di passare il controllo al programma utente, il sistema ripristina la modalità utente riportando a 1 il valore del bit.

La duplice modalità di funzionamento (*dual mode*) consente la protezione del sistema operativo rispetto al comportamento degli utenti e viceversa. Questo livello di protezione si ottiene definendo come **istruzioni privilegiate** le istruzioni di macchina che possono causare danni allo stato del sistema. Poiché la CPU consente l'esecuzione di queste istruzioni soltanto nella modalità di sistema, se si tenta di far eseguire in modalità utente un'istruzione privilegiata, la CPU non la esegue, ma la tratta come un'istruzione illegale inviando un segnale di eccezione al sistema operativo.

Un esempio di istruzione privilegiata è dato dall'istruzione per passare alla modalità kernel. Altri esempi si riferiscono al controllo dell'I/O, alla gestione del timer (*temporizzatore*) e delle interruzioni. Come si vedrà in seguito, vi sono molte istruzioni privilegiate accessorie.

Possiamo ora esaminare il ciclo di vita di un'istruzione e la sua esecuzione in un elaboratore. All'inizio il controllo appartiene al sistema operativo, dove le istruzioni sono eseguite in modalità di sistema. Nel momento in cui il controllo è ceduto a un'applicazione utente si entra in modalità utente. Alla fine, il controllo è restituito al sistema operativo mediante un'interruzione, un'eccezione o una chiamata di sistema.

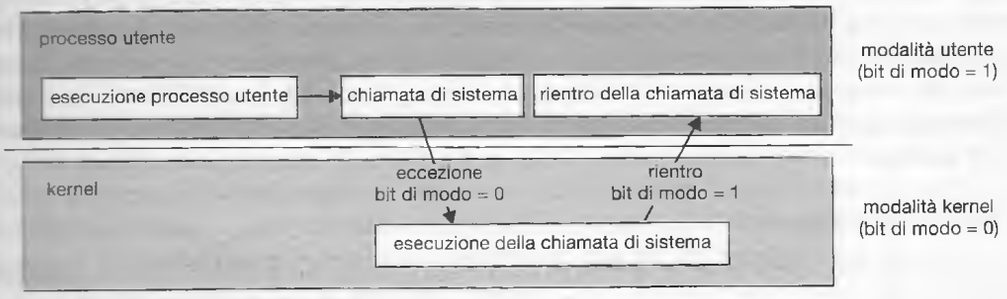

Figura 1.10 Transizione da modalità utente a modalità kernel.

Le chiamate di sistema sono gli strumenti con cui un programma utente richiede al sistema operativo di compiere operazioni a esso riservate, per conto del programma utente. Vi sono vari modi per generare una chiamata di sistema, a seconda delle funzionalità di cui dispone il processore della macchina. In ogni caso, però, le chiamate di sistema sono il mezzo utilizzato dai processi per sollecitare all'azione il sistema operativo. Una chiamata di sistema è solitamente realizzata come un'eccezione che rimanda a un indirizzo specifico nel vettore delle interruzioni. A tale eccezione si può dare esecuzione con un'istruzione `trap` generica, sebbene alcuni sistemi (come quelli appartenenti alla famiglia MIPS R2000) abbiano un'istruzione dedicata `syscall`.

Quando un programma utente esegue una chiamata di sistema, questa è gestita dalla CPU come un'interruzione. Il controllo passa, tramite il vettore delle interruzioni, all'apposita procedura di servizio presente all'interno del sistema operativo e si pone il bit di modo in modalità di sistema. La procedura di servizio della chiamata di sistema è parte integrante del sistema operativo. Il sistema esamina l'istruzione che ha causato l'eccezione, al fine di stabilire la natura della chiamata di sistema, mentre un parametro di tale istruzione definisce il tipo di servizio richiesto dal programma utente. Ulteriori informazioni indispensabili per il completamento della richiesta si possono copiare nei registri, sulla pila o direttamente nella memoria centrale (in locazioni il cui indirizzo è memorizzato nei registri). Il sistema verifica correttezza e legalità dei parametri, soddisfa la richiesta e restituisce il controllo dell'esecuzione all'istruzione immediatamente seguente alla chiamata di sistema. Il Paragrafo 2.3 dedica un ampio approfondimento al concetto di chiamata di sistema.

Se la CPU non dispone di queste due modalità di funzionamento il sistema operativo può andare incontro a serie limitazioni. Il sistema MS-DOS per esempio è stato sviluppato per l'architettura 8088 Intel priva del bit di modo e, quindi, di entrambe le modalità. Un programma utente potrebbe cancellare il sistema operativo scrivendo nuove informazioni nelle locazioni in cui questo è memorizzato, e più programmi potrebbero scrivere contemporaneamente nello stesso dispositivo, con l'eventualità di ottenere risultati disastrosi. Versioni più recenti e progredite delle CPU Intel, come il Pentium, sono dotate della duplice modalità di funzionamento. I più recenti sistemi operativi sviluppati per tali architetture, come Microsoft Windows 2000, Windows XP e i sistemi Solaris x86 traggono vantaggio da questa caratteristica e garantiscono una maggiore protezione del sistema operativo.

Una volta che la protezione hardware sia attiva, gli errori di violazione della modalità sono rilevati dall'hardware stesso, e di norma sono gestiti dal sistema operativo. Se un programma utente causa un errore (per esempio tentando di eseguire un'istruzione illegale o di accedere a una zona di memoria che esula dallo spazio degli indirizzi dell'utente) l'hardware reagirà sollevando un'eccezione. L'eccezione cede il controllo, attraverso il vettore delle interruzioni, al sistema operativo, cioè segue una condotta analoga a quanto farebbe un'interruzione. Quando si imbatte nell'errore di un programma, il sistema operativo deve terminare il programma in maniera anomala. La cosa è gestita dal medesimo codice preposto alla terminazione anomala di un processo a seguito di una richiesta dell'utente: si compone un messaggio di errore appropriato, e si rilascia la memoria del programma incriminato. Il contenuto della memoria rilasciata è solitamente trascritto su un file, perché l'utente o il programmatore possano esaminarlo ed eventualmente correggerlo in vista di un nuovo utilizzo del programma.

1.5.2 Timer

Occorre assicurare che il sistema operativo mantenga il controllo dell'elaborazione, cioè impedire che un programma utente entri in un ciclo infinito o non richieda servizi del sistema

senza più restituire il controllo al sistema operativo. A tale scopo si può usare un timer, programmabile affinché invii un segnale d'interruzione alla CPU a intervalli di tempo specificati, che possono essere fissi (per esempio, di 1/60 di secondo) o variabili (per esempio, da un millisecondo a un secondo). Un **timer variabile** di solito si realizza mediante un generatore di impulsi a frequenza fissa e un contatore. Il sistema operativo assegna un valore al contatore, che si decrementa a ogni impulso e quando raggiunge il valore 0 si genera un segnale d'interruzione. Per esempio, un contatore di 10 bit con un generatore di impulsi con periodo di 1 millisecondo consente la generazione di interruzioni a intervalli compresi tra 1 e 1024 millisecondi, con incrementi di 1 millisecondo.

Prima di restituire all'utente il controllo dell'esecuzione, il sistema assegna un valore al timer. Se esso esaurisce questo intervallo genera un'interruzione che causa il trasferimento del controllo al sistema operativo, che può decidere se gestire le interruzioni come un errore fatale o concedere altro tempo al programma. Ovviamente, anche le istruzioni usate dal sistema per modificare il funzionamento del timer si possono eseguire soltanto in modalità di sistema.

La presenza di un timer garantisce quindi che nessun programma utente possa essere eseguito troppo a lungo. Una tecnica semplice consiste nell'impostare un contatore con un valore pari al tempo concesso al programma per la propria esecuzione. Per esempio, se il programma richiede 7 minuti si dovrebbe impostare il contatore al valore 420. Il timer genererà un'interruzione ogni secondo e il contatore sarà decrementato di 1; fintanto che il valore resta positivo, il controllo ritorna al programma utente; quando il contatore raggiunge un valore negativo, il sistema operativo termina l'esecuzione del programma per il superamento del tempo a esso assegnato.

1.6 Gestione dei processi

Un programma fa qualcosa soltanto se la CPU esegue le istruzioni che lo costituiscono. Benché la sua definizione sia più generale, un processo d'elaborazione o, più brevemente, processo, si può in prima istanza considerare come un programma in esecuzione. Un programma utente, come un compilatore, eseguito in un ambiente a partizione del tempo d'elaborazione, è un processo; un programma d'elaborazione di testi eseguito da un PC per un singolo utente è un processo; così come lo è un servizio di sistema, per esempio l'invio di dati a una stampante. Per il momento ci si può limitare a considerare un processo come un lavoro d'elaborazione (*job*) o un programma eseguito in un ambiente a partizione del tempo d'elaborazione, anche se il concetto è più generale. Come si descrive nel Capitolo 3, si possono avere chiamate di sistema che permettono ai processi di creare sottoprocessi da eseguire in modo concorrente.

Per svolgere i propri compiti, un processo necessita di alcune risorse, tra cui tempo di CPU, memoria, file e dispositivi di I/O. Queste risorse si possono attribuire al processo al momento della sua creazione, oppure si possono assegnare durante l'esecuzione. Oltre alle diverse risorse fisiche e logiche assegnate a un processo durante la sua creazione, si possono considerare anche alcuni dati d'inizializzazione da passare di volta in volta al processo stesso. Per esempio, un processo che ha lo scopo di mostrare su uno schermo lo stato di un file, riceve il nome del file ed esegue le appropriate istruzioni e chiamate di sistema che gli consentono di ottenere e mostrare sullo schermo le informazioni desiderate; quando il processo termina, il sistema operativo riprende il controllo delle risorse impiegate dal processo.

Un programma di per sé *non* è un processo d'elaborazione; un programma è un'entità *passiva*, come il contenuto di un file memorizzato in un disco, mentre un processo è un'en-

tità *attiva*, con un **contatore di programma** che indica la successiva istruzione da eseguire (i thread sono trattati nel Capitolo 4). L'esecuzione di un processo deve essere sequenziale: la CPU esegue le istruzioni del processo una dopo l'altra, finché il processo termina; inoltre in ogni istante si esegue al massimo un'istruzione del processo. Quindi, anche se due processi possono corrispondere allo stesso programma, si considerano in ogni caso due sequenze d'esecuzione separate. Un processo multithread possiede più contatori di programma, ognuno dei quali punta all'istruzione successiva da eseguire per un dato thread. Il processo è l'unità di lavoro di un sistema. Tale sistema è costituito di un gruppo di processi, alcuni dei quali del sistema operativo (che eseguono codice di sistema), mentre altri sono processi utenti (che eseguono codice utente). Tutti questi processi si possono potenzialmente eseguire in modo concorrente, avvicendandosi nell'uso della CPU.

Il sistema operativo è responsabile delle seguenti attività connesse alla gestione dei processi:

- creazione e cancellazione dei processi utenti e di sistema;
- sospensione e ripristino dei processi;
- fornitura di meccanismi per la sincronizzazione dei processi;
- fornitura di meccanismi per la comunicazione tra processi;
- fornitura di meccanismi per la gestione delle situazioni di stallo (*deadlock*).

Le tecniche di gestione dei processi sono trattate in modo approfondito nei Capitoli dal 3 al 6.

1.7 Gestione della memoria

Come si è accennato nel Paragrafo 1.2.2, la memoria centrale è fondamentale per il funzionamento di un moderno sistema di calcolo. Si tratta di un vasto vettore di dimensioni che variano tra le centinaia di migliaia e i miliardi di parole, ciascuna delle quali è dotata del proprio indirizzo. È un magazzino di dati velocemente accessibile ed è condivisa dalla CPU e da alcuni dispositivi di I/O. La CPU legge le istruzioni dalla memoria centrale durante il ciclo di prelievo delle istruzioni, oltre a leggere e scrivere i dati nella memoria centrale durante il ciclo d'accesso ai dati (su di un'architettura Von Neumann). Anche il funzionamento dell'I/O realizzato col DMA legge e scrive dati nella memoria centrale. Generalmente la memoria centrale è l'unico ampio dispositivo di memorizzazione a cui la CPU può far riferimento e accedere in modo diretto. Per esempio, affinché la CPU possa gestire i dati di un disco, occorre che essi siano prima trasferiti nella memoria centrale attraverso le richieste di I/O generate dalla CPU. In modo analogo, la CPU può eseguire le istruzioni solo se si trovano nella memoria.

Per eseguire un programma è necessario che questo sia associato a indirizzi assoluti e sia caricato nella memoria. Durante l'esecuzione del programma, la CPU accede alle proprie istruzioni e ai dati provenienti dalla memoria, generando i suddetti indirizzi assoluti. Quando il programma termina, si dichiara disponibile il suo spazio di memoria; a questo punto si può caricare ed eseguire il programma successivo.

Per migliorare l'utilizzo della CPU e la rapidità con la quale il calcolatore risponde ai propri utenti i computer a uso generale devono tenere molti programmi in memoria. Esistono diversi schemi di gestione della memoria; l'efficacia di ogni algoritmo dipende dalla situazione specifica. Poiché ogni algoritmo richiede specifiche caratteristiche, la scelta di un particolare schema di gestione della memoria dipende da molti fattori, principalmente dal tipo di *architettura* del sistema.

Il sistema operativo è responsabile delle seguenti attività connesse alla gestione della memoria centrale:

- tenere traccia di quali parti di memoria sono attualmente usate e da chi;
- decidere quali processi (o parti di essi) e dati debbano essere caricati in memoria o trasferiti altrove;
- assegnare e revocare lo spazio di memoria secondo le necessità.

Le tecniche di gestione della memoria sono trattate in profondità nei Capitoli 8 e 9.

1.8 Gestione della memoria di massa

Per facilitare gli utenti, il sistema operativo fornisce un'interfaccia logica uniforme per la memorizzazione delle informazioni. Esso, cioè, prescinde dalle caratteristiche fisiche dei dispositivi di memorizzazione, definendo un'unità logica di archiviazione, il **file**. Il sistema operativo associa i file a supporti fisici, e vi accede tramite i dispositivi di memorizzazione delle informazioni.

1.8.1 Gestione dei file

La gestione dei file è uno dei componenti più visibili di un sistema operativo. I calcolatori possono registrare le informazioni su molti mezzi fisici diversi; i più diffusi sono il nastro magnetico, il disco magnetico e il disco ottico. Ciascuno ha caratteristiche proprie e una propria organizzazione fisica, ed è controllato da un dispositivo, come un'unità a disco o a nastro, avente anch'esso caratteristiche proprie: velocità, capacità, rapidità nel trasferimento dei dati, metodi d'accesso (diretto o sequenziale).

Un **file** è una raccolta d'informazioni correlate definite dal loro creatore. Comunemente, i file rappresentano programmi, sia sorgente sia oggetto, e dati. I file di dati possono essere numerici, alfabetici, alfanumerici o binari; la loro forma può essere libera, come nei file di testo, oppure rigidamente formattata, per esempio in campi fissi. Il concetto di file è chiaramente molto generale.

Il sistema operativo realizza il concetto astratto di file gestendo i mezzi di memoria di massa, come nastri e dischi, e i dispositivi che li controllano. I file sono generalmente organizzati in directory, che ne facilitano l'uso. Infine, se più utenti hanno accesso ai file, si potrebbe voler controllare chi ha la possibilità di accedervi e in che modo (per esempio, lettura, scrittura, aggiunta).

Il sistema operativo è responsabile delle seguenti attività connesse alla gestione dei file:

- creazione e cancellazione di file;
- creazione e cancellazione di directory;
- fornitura delle funzioni fondamentali per la gestione di file e directory;
- associazione dei file ai dispositivi di memoria secondaria;
- creazione di copie di riserva (*backup*) dei file su dispositivi di memorizzazione non volatili.

Le tecniche di gestione dei file sono trattate nei Capitoli 10 e 11.

1.8.2 Gestione della memoria di massa

Lo scopo principale di un sistema di calcolo è quello di eseguire programmi. Durante l'esecuzione, i programmi, insieme con i dati cui accedono, devono trovarsi nella memoria centrale. Giacché la memoria centrale è troppo piccola per contenere tutti i dati e tutti i programmi, e il suo contenuto va perduto se il sistema si spegne, il calcolatore deve disporre di una memoria secondaria a sostegno della memoria centrale. La maggior parte dei moderni sistemi di calcolo impiega i dischi come principale mezzo di memorizzazione secondaria, sia per i programmi sia per i dati. I dischi contengono la maggior parte dei programmi, compresi i compilatori, gli assemblatori, le procedure di ordinamento e gli editor. Questi programmi rimangono nel disco fino al momento del caricamento in memoria e si servono del disco sia come sorgente sia come destinazione delle loro computazioni; per tale ragione è fondamentale che la registrazione nei dischi sia gestita correttamente. Il sistema operativo è responsabile delle seguenti attività connesse alla gestione dei dischi:

- gestione dello spazio libero;
- assegnazione dello spazio;
- scheduling del disco.

Il frequente uso della memoria secondaria impone una sua gestione efficiente. Infatti, l'efficienza complessiva di un calcolatore può dipendere dalla velocità del sottosistema di gestione dei dischi e dagli algoritmi che lo gestiscono.

Vi sono però svariati frangenti in cui tornano utili dispositivi di memorizzazione più lenti, meno costosi e talora più capienti della memoria secondaria. Le copie di riserva dei dischi di sistema, i dati usati raramente e gli archivi a lungo termine sono alcuni esempi. Le unità a nastro magnetico, i CD e i DVD, sono tipici dispositivi di **memoria terziaria.** I media (nastri e i dischi ottici) comprendono i formati monoscrivibili WORM (*write once, read many times*) e riscrivibili RW (*read and write*).

La memoria terziaria ha scarso impatto sulle prestazioni del sistema, ma deve pur essere gestita. Alcuni sistemi si assumono tale onere direttamente, mentre altri lo delegano a programmi applicativi. Tra le funzioni che i sistemi possono fornire citiamo l'installazione e la rimozione dei media dei dispositivi, l'allocazione dei dispositivi ai processi che ne richiedano l'uso esclusivo, e il trasferimento alla memoria terziaria dei dati provenienti da quella secondaria.

Le tecniche di gestione della memoria secondaria e terziaria saranno approfondite nel Capitolo 12.

1.8.3 Cache

Il concetto di cache è un principio importante di un sistema di calcolo. Di norma le informazioni sono mantenute in unità di memoria come la memoria centrale; al momento del loro uso si copiano temporaneamente in un'unità più veloce: la cache. Quando si deve accedere a una particolare informazione, innanzitutto si controlla se è già presente all'interno della cache; in tal caso si adopera direttamente la copia contenuta nella cache, altrimenti la si preleva dalla memoria centrale e la si copia nella cache, poiché si suppone che questa informazione presto servirà ancora.

Inoltre, i registri programmabili presenti all'interno della CPU, come i registri indice, rappresentano per la memoria centrale una cache ad alta velocità. Il programmatore (o il compilatore) codifica gli algoritmi di assegnazione e aggiornamento dei registri in modo da stabilire quali informazioni mantenere nei registri e quali nella memoria centrale. Esistono anche cache che sono interamente gestite dall'architettura del sistema: la maggior parte dei

sistemi è dotata per esempio di una cache per la memorizzazione delle istruzioni che presumibilmente saranno eseguite dopo l'istruzione corrente. Senza di essa, la CPU dovrebbe attendere parecchi cicli prima che un'istruzione sia prelevata dalla memoria. Per simili motivi, la maggior parte dei sistemi è dotata di una o più cache di dati nella gerarchia delle memorie. In questo testo non ci si occupa di tali dispositivi, poiché sono componenti non controllabili dal sistema operativo.

Data la capacità limitata di questi dispositivi, la **gestione della cache** è un importante problema di progettazione. Da un'attenta selezione delle dimensioni e dei criteri di aggiornamento della cache può conseguire un notevole incremento delle prestazioni del sistema. La Figura 1.11 presenta un confronto tra le prestazioni di vari dispositivi per la memorizzazione in piccoli server e potenti stazioni di lavoro. La necessità delle memorie cache emerge chiaramente. Nel Capitolo 9 si discutono alcuni algoritmi per la sostituzione degli elementi contenuti nelle cache controllabili dal sistema operativo.

La memoria centrale si può considerare una cache per la memoria secondaria, giacché i dati in essa contenuti devono essere riportati nella memoria centrale per poter essere usati e qui devono risiedere prima di essere trasferiti nella memoria secondaria. I dati del file system, che risiedono in modo permanente nella memoria secondaria, possono apparire a diversi livelli della gerarchia di memoria. A livello più alto, il sistema operativo può mantenere una cache dei dati del file system in memoria centrale. Anche i dischi RAM si possono impiegare come veloci sistemi di memoria volatile cui si accede tramite l'interfaccia del file system. La memorizzazione secondaria si effettua generalmente in dischi magnetici; copie di riserva del loro contenuto spesso si registrano in nastri magnetici o dischi rimovibili. Alcuni sistemi, per ridurre i costi di memorizzazione, archiviano automaticamente i vecchi file di dati trasferendoli dalla memoria secondaria alla memoria terziaria, costituita per esempio da *juke-box* di nastri (Capitolo 12).

Il movimento delle informazioni tra i vari livelli della gerarchia può essere quindi sia implicito sia esplicito, secondo l'architettura e il sistema operativo che la controlla. Per esempio, il trasferimento dei dati tra la cache e i registri della CPU è di solito svolto dall'architettura del sistema senza alcun intervento del sistema operativo. Diversamente, il trasferimento dei dati dai dischi alla memoria è di solito gestito dal sistema operativo.

Livello	1	2	3	4
Nome	registri	cache	memoria centrale	disco
Dimensione tipica	< 1 KB	< 16 MB	< 16 GB	> 100 GB
Tecnologia	memoria dedicata con porte multiple (CMOS)	CMOS SRAM (on-chip o off-chip)	CMOS DRAM	disco magnetico
Tempo d'accesso (ns)	0,25 – 0,5	0,5 – 25	80 – 250	5.000,000
Ampiezza di banda (MB/s)	20.000 – 100.000	5000 – 10.000	1000 – 5000	20 – 150
Gestito da	compilatore	hardware	sistema operativo	sistema operativo
Supportato da	cache	memoria centrale	disco	CD o nastro

Figura 1.11 Prestazioni relative a varie forme di archiviazione dei dati.

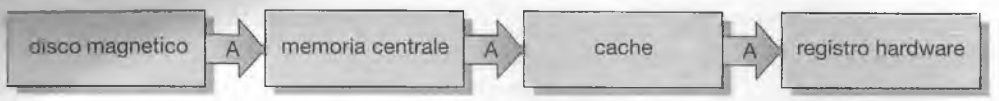

Figura 1.12 Migrazione di un intero *A* da un disco a un registro.

In una struttura gerarchica come quella appena introdotta, può accadere che gli stessi dati siano mantenuti contemporaneamente in diversi livelli del sistema di memorizzazione. Per esempio, si supponga di incrementare di 1 il valore di un numero intero A contenuto in un file B, registrato in un disco magnetico. L'operazione d'incremento prevede innanzitutto l'esecuzione di un'operazione di I/O per copiare nella memoria centrale il blocco di disco contenente il valore di A. Questa operazione è di solito seguita dalla copiatura di A all'interno della cache, e di qui in uno dei registri interni della CPU. Quindi, esistono diverse copie di A memorizzate nei vari dispositivi coinvolti nel trasferimento: nel disco magnetico, nella memoria centrale, nella cache e in un registro interno della CPU (Figura 1.12). Dopo l'incremento del valore della copia contenuta nel registro interno, il valore di A sarà diverso da quello assunto dalle altre copie della stessa variabile. Le diverse copie di A avranno lo stesso valore solamente dopo che il nuovo valore sarà stato riportato dal registro interno della CPU alla copia di A residente nel disco.

In un ambiente di calcolo che ammette l'esecuzione di un solo processo alla volta, questo modo di operare non pone particolari difficoltà, poiché ogni accesso ad A coinvolge la copia posta al livello più alto della gerarchia. Viceversa, nei sistemi a partizione del tempo d'elaborazione, in cui il controllo della CPU passa da un processo all'altro, è necessario prestare una particolare attenzione al fine di assicurare che qualsiasi processo che desideri accedere ad A ottenga dal sistema il valore della variabile aggiornato più di recente.

La situazione si complica ulteriormente negli ambienti multiprocessore, dove ciascuna di esse, oltre ai registri interni, contiene anche una cache locale. Nei sistemi del genere possono esistere più copie simultanee di A mantenute in cache differenti. Poiché le diverse unità d'elaborazione possono operare in modo concorrente, è necessario che l'aggiornamento del valore di una qualsiasi delle copie di A contenute nelle cache locali si rifletta immediatamente in tutte le cache in cui A risiede. Questa situazione, nota come **coerenza della cache**, di solito si risolve a livello dell'architettura del sistema, quindi a un livello più basso di quello del sistema operativo.

In un sistema distribuito la situazione diventa ancora più complessa, poiché può accadere che più copie (chiamate in questo caso repliche) dello stesso file siano mantenute in differenti calcolatori, dislocati in luoghi fisici diversi. Essendoci la possibilità di accedere e modificare in modo concorrente queste repliche, è necessario fare in modo che ogni modifica a una qualunque replica si rifletta quanto prima sulle altre. Nel Capitolo 17 si discutono diversi metodi che consentono di soddisfare questo requisito.

1.8.4 Sistemi di I/O

Uno degli scopi di un sistema operativo è nascondere all'utente le caratteristiche degli specifici dispositivi. In UNIX, per esempio, le caratteristiche dei dispositivi di I/O sono nascoste alla maggior parte dello stesso sistema operativo dal **sottosistema di I/O**, che è composto delle seguenti parti:

- un componente di gestione della memoria comprendente la gestione delle regioni della memoria riservate ai trasferimenti di I/O (*buffer*), la gestione delle cache e la gestione asincrona delle operazioni di I/O e dell'esecuzione di più processi (*spooling*);
- un'interfaccia generale per i driver dei dispositivi;
- i driver per gli specifici dispositivi.

Soltanto il driver del dispositivo conosce le caratteristiche dello specifico dispositivo cui è assegnato.

Nel Paragrafo 1.2.3 è stato presentato l'uso dei gestori dei segnali d'interruzione e dei driver dei dispositivi per la costruzione di sottosistemi di I/O efficienti. Nel Capitolo 13 si discute per esteso il modo in cui il sottosistema di I/O interagisce con gli altri componenti del sistema, come gestisce i dispositivi, trasferisce i dati e individua il completamento delle operazioni di I/O.

1.9 Protezione e sicurezza

Se diversi utenti usufruiscono dello stesso elaboratore che consente la simultanea esecuzione di processi multipli, l'accesso ai dati dovrà essere disciplinato da regole. Queste varranno ad assicurare che i file, i segmenti di memoria, la CPU e le altre risorse possano essere manipolati solo dai processi che abbiano ottenuto apposita autorizzazione dal sistema operativo. Per esempio, il dispositivo fisico per l'indirizzamento della memoria garantisce il fatto che un processo sia eseguito solo entro il proprio spazio degli indirizzi. Il timer impedisce che un processo, dopo aver conquistato il controllo della CPU, non lo rimetta più al sistema operativo. I registri di controllo dei dispositivi non sono accessibili agli utenti, quindi l'integrità delle varie periferiche viene protetta.

Per **protezione**, quindi, si intende ciascuna strategia di controllo dell'accesso alle risorse possedute da un elaboratore, da parte di processi o utenti. Le strategie di protezione devono fornire le specifiche dei controlli da attuare e gli strumenti per la loro effettiva applicazione.

La protezione può migliorare l'affidabilità rilevando errori nascosti nelle interfacce tra i componenti dei sottosistemi. Il beneficio che spesso deriva da una tempestiva rilevazione degli errori nelle interfacce è di prevenire la contaminazione di un sottosistema sano a opera di un altro sottosistema infetto. Non proteggere una risorsa significa lasciare via libera al suo utilizzo (o all'utilizzo scorretto) da parte di utenti incompetenti o non autorizzati. Un sistema informato da strategie di protezione offre i mezzi per distinguere l'uso autorizzato da quello non autorizzato, come si vedrà nel Capitolo 14.

Pur essendo dotato di protezione adeguata, un sistema può rimanere esposto agli accessi abusivi, rischiando malfunzionamenti. Si consideri un'utente le cui le informazioni di autenticazione siano state trafugate. I suoi dati rischiano di essere copiati o cancellati, anche se è attiva la protezione dei file e della memoria. È compito della **sicurezza** difendere il sistema da attacchi provenienti dall'interno o dall'esterno. La varietà di minacce conosciute è enorme: si va dai virus e i worm, agli attacchi che paralizzano il servizio (appropriandosi di tutte le risorse del sistema e dunque escludendo da esso i legittimi utenti), al furto d'identità e la sottrazione del servizio (uso non autorizzato di un sistema). Per alcuni sistemi si può ritenere che l'attività di prevenzione da attacchi siffatti appartenga a una funzione del sistema, mentre ve ne sono altri che delegano la difesa a regole esterne o a programmi aggiuntivi. A causa dell'allarmante incremento di incidenti legati alla sicurezza, le problematiche della sicurezza del sistema operativo si collocano in un settore che vede crescere rapidamente la ricerca e la sperimentazione. La sicurezza è analizzata nel Capitolo 15.

La protezione e la sicurezza presuppongono che il sistema sia in grado di distinguere tra tutti i propri utenti. Nella maggior parte dei sistemi operativi è disponibile un elenco di nomi degli utenti e dei loro **identificatori utente** (**user** ID). Nel gergo di Windows NT, si parla di ID **di sicurezza** (SID). Si tratta di ID numerici che identificano univocamente l'utente. Quando un utente si collega al sistema, la fase di autenticazione determina l'ID utente corretto. Tale ID utente è associato a tutti i processi e i thread del soggetto in questione. Se l'utente ha necessità di leggere un ID numerico, il sistema lo riconverte in forma di nome utente grazie al suo elenco dei nomi degli utenti.

In certe circostanze è preferibile distinguere tra gruppi di utenti invece che tra utenti singoli. Per esempio, il proprietario di un file su un sistema UNIX potrebbe aver titolo a effettuare qualsiasi operazione su quel file, mentre un gruppo selezionato di utenti può essere abilitato soltanto alla lettura del file. Per ottenere questo, dobbiamo attribuire un nome al gruppo e identificare gli utenti che vi appartengono. La funzionalità è realizzabile creando un elenco esaustivo dei nomi dei gruppi e dei relativi **identificatori** di gruppo. Un utente può fare parte di uno o più gruppi, a seconda delle scelte compiute in sede di progettazione del sistema operativo. L'identificatore di gruppo è incluso in tutti i processi e i thread a esso relativi.

Durante il normale utilizzo del sistema, all'utente sono sufficienti un identificatore utente e un identificatore del gruppo. Tuttavia, gli utenti devono talvolta *conquistare privilegi*, ossia ottenere permessi ausiliari per certe attività; per esempio, un dato utente potrebbe aver bisogno di accedere a un dispositivo il cui uso è riservato. Vi sono vari metodi che permettono agli utenti di conquistare privilegi. In UNIX, per esempio, se è presente l'attributo `setuid` in un programma, esso potrà essere eseguito con l'identificatore utente del proprietario del file, piuttosto che con quello dell'utente reale. Il processo esegue con questo identificatore effettivo finché non rinunci a questo privilegio, o termini.

1.10 Sistemi distribuiti

Per sistema distribuito si intende un insieme di elaboratori collocati a distanza, e con caratteristiche spesso eterogenee, interconnessi da una rete di calcolatori per consentire agli utenti l'accesso alle varie risorse dei singoli sistemi. L'accesso a una risorsa condivisa aumenta la velocità di calcolo, la funzionalità, la disponibilità dei dati e il grado di affidabilità. Alcuni sistemi operativi paragonano l'accesso alla rete a una forma di accesso ai file, demandandone i dettagli dell'accesso alla rete al driver dell'interfaccia fisica con la rete. Altri sistemi, invece, permettono agli utenti di invocare funzioni specifiche della rete. Generalmente si riscontra nei sistemi una combinazione delle due modalità: per esempio FTP e NFS. I protocolli che danno vita a un sistema distribuito possono determinarne l'utilità e l'apprezzamento in misura considerevole.

Una **rete** si può considerare, in parole semplici, come un canale di comunicazione tra due o più sistemi. I sistemi distribuiti si basano sulle reti per realizzare le proprie funzioni, sfruttano la capacità di comunicazione per cooperare nella soluzione dei problemi di calcolo e per fornire agli utenti un ricco insieme di funzioni. Le reti differiscono per i protocolli usati, per le distanze tra i nodi e per il mezzo attraverso il quale avviene la comunicazione. Sebbene siano largamente usati sia il protocollo ATM sia altri protocolli, il più diffuso protocollo di comunicazione è il TCP/IP. Anche la disponibilità dei protocolli di rete è piuttosto varia nei diversi sistemi operativi. La maggior parte di essi, inclusi i sistemi Windows e UNIX, impiega il TCP/IP. Alcuni sistemi dispongono di propri protocolli che soddisfano esigenze specifiche. Affinché un sistema operativo possa gestire un protocollo di rete è necessa-

ria la presenza di un dispositivo d'interfaccia – un adattatore di rete, per esempio – con un driver per gestirlo, oltre al software per la gestione dei dati.

Le reti si classificano secondo le distanze tra i loro nodi: una **rete locale** (LAN) comprende nodi all'interno della stessa stanza, piano o edificio; una **rete geografica** (WAN) si estende a gruppi di edifici, città, o al territorio di una regione o di uno stato. Una società multinazionale, per esempio, potrebbe disporre di una rete WAN per connettere i propri uffici nel mondo. Queste reti possono funzionare con uno o più protocolli e il continuo sviluppo di nuove tecnologie fa sì che si definiscano nuovi tipi di reti. Le **reti metropolitane** (MAN) per esempio collegano gli edifici di un'intera città; i dispositivi BlueTooth e 802.11 comunicano a breve distanza, dell'ordine delle decine di metri, creando essenzialmente una **microrete** (*small-area network*), simile a quelle domestiche.

I mezzi di trasmissione che s'impiegano nelle reti sono altrettanto vari: fili di rame, fibre ottiche, trasmissioni via satellite, sistemi a microonde e sistemi radio; anche il collegamento dei dispositivi di calcolo ai telefoni cellulari crea una rete, così come, per creare una rete, si può usare anche la capacità di comunicazione a brevissima distanza dei dispositivi a raggi infrarossi. In breve, ogni volta che comunicano, i calcolatori usano o creano reti, che ovviamente si differenziano per prestazioni e affidabilità.

Taluni sistemi operativi hanno interpretato l'idea delle reti e dei sistemi distribuiti secondo un'ottica più vasta rispetto alla semplice fornitura della connettività di rete. Un **sistema operativo di rete** è dotato di caratteristiche quali la condivisione dei file attraverso la rete e di un modello di comunicazione per i processi attivi su elaboratori diversi, che possono così scambiarsi messaggi. Un computer che funzioni con un sistema operativo di rete agisce con una certa autonomia rispetto a tutti gli altri computer della rete, benché sia conscio della presenza della rete e, dunque, possa interagire con gli altri computer che ne fanno parte. Un sistema operativo distribuito offre un ambiente meno autonomo: la stretta comunicazione che si instaura tra i diversi elaboratori partecipanti tende a dare l'impressione che vi sia un solo sistema incaricato di presidiare la rete.

I Capitoli dal 16 al 18 sono dedicati alle reti di calcolatori e ai sistemi distribuiti.

1.11 Sistemi a orientamento specifico

Gli argomenti esaminati finora si riferivano a sistemi di calcolo più familiari, ovvero quelli con finalità generali. Esistono, d'altra parte, varie categorie di sistemi di calcolo, inerenti a particolari aree della computazione, le cui funzioni sono più limitate.

1.11.1 Sistemi integrati real-time

In termini quantitativi, i computer integrati (*embedded computers*) attualmente costituiscono la tipologia predominante di elaboratore. Questi dispositivi si ritrovano dappertutto, dai motori delle auto ai robot industriali, dai videoregistratori ai forni a microonde. Di solito, hanno compiti molto precisi: i sistemi su cui sono impiantati sono spesso rudimentali, per cui offrono funzionalità limitate. Essi presentano un'interfaccia utente scarsamente sviluppata, se non assente, dato che sono spesso concepiti per la sorveglianza e la gestione di dispositivi meccanici, quali i motori delle automobili e i bracci dei robot.

Ciò che contraddistingue i sistemi integrati è la loro grande variabilità. Talvolta possono essere elaboratori di uso generale con sistemi operativi standard – come UNIX – che sfruttano applicazioni create appositamente per implementare una funzionalità. Altri sono dispositivi meccanici che ospitano un sistema operativo integrato a obiettivo specifico, che

consente di ottenere proprio la funzionalità desiderata. Inoltre, esistono dispositivi meccanici che hanno al loro interno circuiti integrati per applicazioni specifiche (ASIC), capaci di svolgere il loro lavoro senza un sistema operativo.

La diffusione dei sistemi integrati è in continua espansione. Indubbiamente, sono destinate a crescere anche le potenzialità applicative di questi congegni, sia come unità indipendenti sia in qualità di membri delle reti e di Internet. È già possibile l'automatizzazione di intere abitazioni, cosicché un computer centrale – sia che si tratti di un computer con finalità generali o di un sistema integrato – sia in grado di controllare il riscaldamento, l'illuminazione o i sistemi di allarme. Attraverso Internet, la propria casa può essere riscaldata a distanza prima di rientrarvi. Un giorno, il frigorifero potrà forse decidere autonomamente di chiamare il droghiere per il rifornimento del latte.

I sistemi integrati funzionano quasi sempre con **sistemi operativi real-time**. Essi si utilizzano quando siano stati imposti rigidi vincoli di tempo a carico del processore o al flusso dei dati; per questa ragione, sono spesso adoperati come dispositivi di controllo per applicazioni dedicate. I sensori trasmettono i dati al computer, che deve analizzarli e, a volte, prendere le misure adatte per il controllo del sistema. I sistemi adibiti al controllo di esperimenti scientifici, i sistemi di controllo industriale e taluni sistemi per la visualizzazione, sono sistemi real-time. Alcuni motori a iniezione di benzina, i meccanismi per il controllo di elettrodomestici e i sistemi di difesa armata sono, anch'essi, real-time.

Questo tipo di sistema ha l'obbligo di rispettare categoricamente i propri vincoli di tempo, che sono definiti con precisione. L'elaborazione *deve* avvenire entro i limiti prestabiliti: in caso contrario, il sistema andrà in crisi. Per esempio, non serve a nulla che il braccio di un robot riceva l'ordine di fermarsi solo *dopo* essersi scontrato con l'automobile che era impegnato a costruire. Le prestazioni di un sistema real-time sono soddisfacenti solo se esso, generando il risultato corretto, rispetta precise scadenze. Si confronti questo tipo di sistema con un sistema a tempo ripartito (*time-sharing system*), in cui è auspicabile una risposta rapida (ma non obbligatoria), o con un sistema a lotti, che potrebbe essere del tutto esente da vincoli di tempo.

Dei sistemi integrati in tempo reale si riparlerà, con dovizia di dettagli, nel Capitolo 19. Il Capitolo 5 esamina la tipologia di scheduling da adottare affinché un sistema operativo possa eseguire applicazioni real-time. La gestione della memoria nell'ambito dell'elaborazione in tempo reale è illustrata nel Capitolo 9. Infine, il Capitolo 22 è dedicato ai componenti real-time del sistema operativo Windows XP.

1.11.2 Sistemi multimediali

Gran parte dei sistemi operativi è stata concepita per la gestione di dati tradizionali, quali file di testo, programmi, documenti di videoscrittura e fogli di calcolo. Tuttavia, per effetto di una recente tendenza della tecnologia, negli elaboratori fanno la loro comparsa sempre più spesso i dati multimediali o dati a trasmissione continua. Essi consistono sia di file audio e video che di file tradizionali. Questi dati si distinguono da quelli tradizionali perché la loro trasmissione all'utente finale deve attenersi a precise frequenze: per esempio, nel caso di un video, 30 fotogrammi al secondo.

Multimediale è sinonimo, al giorno d'oggi, di varie applicazioni che incontrano il gradimento del pubblico. Alcune di esse sono i file audio MP3, i film in DVD, la video-conferenza, i video clip contenenti anteprime cinematografiche o notiziari. Tra le applicazioni multimediali citiamo anche gli eventi trasmessi in diretta sul World Wide Web, relativi a discorsi o gare sportive, e persino le webcam dal vivo con cui uno spettatore a Manhattan può osservare gli avventori di un caffè parigino. Le applicazioni multimediali non devono neces-

sariamente essere o solo audio, o solo video; infatti, un'applicazione multimediale sovente comprende una combinazione dei due. Un film, per esempio, si compone di tracce audio e video separate. Né queste applicazioni sono destinate unicamente ai personal computer; sempre di più, esse sono dirette a prodotti più piccoli quali gli assistenti personali digitali (PDA) e i telefoni cellulari. Un intermediario sui mercati finanziari, a titolo di esempio, potrà seguire l'andamento delle azioni in tempo reale sul proprio PDA, tramite una connessione senza fili.

Nel Capitolo 20 prenderemo in analisi le esigenze delle applicazioni multimediali. Approfondiremo ciò che distingue i dati multimediali da quelli tradizionali, sottolineando l'impatto di tali differenze sulla progettazione di sistemi operativi idonei a soddisfare i requisiti dei sistemi multimediali.

1.11.3 Sistemi palmari

I sistemi palmari (*handheld systems*) comprendono gli assistenti digitali, noti come PDA (*personal digital assistant*), come i computer palmari e tascabili (*palm-pilot*) o i telefoni cellulari, molti dei quali adottano sistemi operativi integrati a orientamento specifico. I progettisti di sistemi e applicazioni per i sistemi palmari devono affrontare molti problemi, alcuni dei quali correlati alle dimensioni di tali dispositivi. I PDA più diffusi sono lunghi circa dodici centimetri, larghi circa otto, e pesano tra cento e duecentocinquanta grammi. A causa delle piccole dimensioni, la maggior parte dei dispositivi palmari dispone di una memoria limitata, unità d'elaborazione lente e schermi piccoli.

La quantità di memoria di molti dispositivi palmari varia a seconda del modello, ma in genere oscilla tra 1 MB e 1 GB: assai meno di quella di un comune PC o stazione di lavoro, che può essere di diversi gigabyte. Per questo motivo, il sistema operativo e le applicazioni devono gestire la memoria in modo efficiente. Ciò implica, tra l'altro, restituire al gestore della memoria l'intera memoria assegnata, una volta che questa non è più usata. Il Capitolo 9 tratta le tecniche di memoria virtuale, che permettono ai programmatori di scrivere programmi che si comportano come se il sistema avesse più memoria di quella fisicamente disponibile. Attualmente, molti dispositivi palmari non impiegano tecniche di memoria virtuale, quindi impongono ai programmatori di lavorare entro i limiti della memoria fisica.

Un secondo aspetto che riguarda i programmatori di questi dispositivi è la velocità delle CPU, che di solito è solo una frazione di quella di una CPU per PC. D'altra parte, le CPU più veloci consumano più energia, quindi richiederebbero l'impiego di cluster di dimensioni maggiori e la necessità di cambiarle (o ricaricarle) più spesso. Per ridurre al minimo le dimensioni dei dispositivi palmari si usano di solito CPU più piccole, più lente e che consumano meno energia. Quindi, il sistema operativo e le applicazioni si devono progettare in modo da non gravare eccessivamente la CPU.

Un ulteriore problema da affrontare da parte degli sviluppatori di questi dispositivi riguarda le questioni di I/O: a causa dello spazio esiguo, i metodi per l'inserimento dei dati sono limitati all'uso di piccole tastiere, al riconoscimento della scrittura, o a ridotte tastiere simulate sullo schermo. Poiché lo schermo è di modeste dimensioni, le opzioni disponibili per la rappresentazione dei dati in uscita sono limitate. Mentre nei PC sono comuni schermi di 30 pollici, quelli dei dispositivi palmari di solito non superano i 6 × 9 centimetri. Attività comuni come la lettura della posta elettronica o la consultazione del Web si devono condensare in schermi molto piccoli. Un metodo per mostrare il contenuto di pagine web è il servizio di *ritagli di pagine* (**web clipping**), che prevede l'invio di sottoinsiemi delle pagine da mostrare nel dispositivo palmare.

Alcuni dispositivi palmari possono servirsi di tecnologie di comunicazione senza fili, come il sistema BlueTooth o 802.11, che consentono l'accesso al servizio di posta elettronica e al Web. I telefoni cellulari che permettono la connessione alla rete Internet appartengono a questa categoria. Tuttavia, molti PDA attualmente non prevedono l'accesso senza fili; i dati da trasferire a questi dispositivi di solito si trasmettono prima a un PC o a una stazione di lavoro e successivamente al PDA. Alcuni permettono la copiatura diretta dei dati tra dispositivi per mezzo di una connessione a raggi infrarossi.

Generalmente, l'utilità e la portabilità dei PDA compensano i limiti delle loro funzioni. La loro diffusione è in continua espansione rispetto alla connessione alle reti e la disponibilità di altri accessori – macchine fotografiche, lettori MP3, ecc. – li rendono sempre più utili.

1.12 Ambienti d'elaborazione

Finora è stata offerta una panoramica dell'organizzazione dei sistemi di elaborazione e dei principali componenti di un sistema operativo. Il capitolo si conclude con una breve introduzione del loro uso in ambienti d'elaborazione diversi.

1.12.1 Elaborazione tradizionale

Con l'evoluzione delle tecniche d'elaborazione, i confini tra molti ambienti d'elaborazione tradizionale diventano sempre più sfumati. Si consideri per esempio un tipico ambiente d'ufficio:, solo pochi anni fa consisteva di PC connessi in rete, con server che fornivano servizi di accesso ai file e di stampa. L'accesso remoto era difficoltoso e la portabilità si otteneva grazie ai calcolatori portatili che permettevano il trasferimento di una parte dello spazio di lavoro dell'utente. I terminali connessi ai mainframe erano ancora i più diffusi nelle grandi aziende, con ancora meno funzioni d'accesso remoto e di portabilità.

Attualmente si tende ad avere più modi d'accesso a questi ambienti; le tecnologie del Web stanno estendendo i confini del calcolo tradizionale, le aziende realizzano i cosiddetti **portali** che permettono l'accesso tramite il Web ai propri server interni. I **calcolatori di rete** sono essenzialmente terminali adatti all'elaborazione basata sul Web. I sistemi palmari possono sincronizzarsi con i PC per consentire un uso estremamente portatile delle informazioni aziendali; i PDA permettono anche la connessione a **reti senza fili** per accedere al portale web dell'azienda (e alle tantissime altre risorse del Web).

In casa, la maggior parte degli utenti aveva un solo calcolatore con una lenta connessione via modem all'ufficio, alla rete Internet, o a entrambi. Le connessioni di rete veloci, un tempo possibili a costi molto alti, sono ora disponibili a prezzi abbastanza contenuti e permettono l'accesso a maggiori quantità di dati. Queste connessioni veloci consentono ai calcolatori di casa di trasformarsi in server web e di formare reti con stampanti, PC client e server. Alcuni ambienti d'elaborazione domestici sono dotati anche di **firewall** (*barriere antintrusione*) che proteggono dagli attacchi informatici esterni; si tratta di sistemi che solo qualche anno fa costavano migliaia di euro e dieci anni fa nemmeno esistevano.

Nella seconda metà del '900 le risorse elettroniche di calcolo erano scarse (e prima ancora, non esistevano!). C'è stato un periodo di tempo in cui i sistemi erano o a lotti oppure interattivi. I sistemi a lotti elaboravano i processi all'ingrosso, per così dire, con un input predeterminato (da file o da altre fonti). I sistemi interattivi aspettavano di ricevere i dati in ingresso dagli utenti. Per ottenere la massima resa dall'elaboratore, diversi utenti si alternavano su questi sistemi. I sistemi a tempo ripartito impiegavano un timer e algoritmi di sche-

duling per assegnare rapidamente i processi alla CPU, attribuendo a ogni utente una parte delle risorse.

Attualmente, i tradizionali sistemi a tempo ripartito sono rari. La corrispondente strategia di scheduling è tuttora usata dai server e nelle stazioni di lavoro, ma spesso tutti i processi fanno capo allo stesso utente (o a un utente singolo e al sistema operativo). I processi dell'utente, e i processi del sistema che forniscono servizi all'utente, sono gestiti in modo da ricevere entrambi, frequentemente, una fetta del tempo a disposizione. Ciò si può notare, per esempio, osservando le finestre esistenti durante il lavoro di un utente al calcolatore, e notando che molte fra loro possono eseguire nel contempo operazioni diverse.

1.12.2 Computazione client-server

Di pari passo all'aumento di velocità, potenza ed economicità dei PC, i progettisti hanno abbandonato il modello di architettura centralizzata per i sistemi. I PC stanno prendendo il posto dei terminali connessi ai grandi sistemi centralizzati. A riprova di questo fenomeno, accade sempre più spesso che i PC gestiscano in proprio la funzionalità d'interfaccia con l'utente, di cui prima si occupavano i sistemi a livello centrale. Ecco perché molti sistemi odierni fungono da sistemi server per le richieste che ricevono dai sistemi client. Questa variante specialistica dei sistemi distribuiti, che prende il nome di sistema client-server, ha la struttura generale rappresentata nella Figura 1.13.

Schematicamente possiamo suddividere i sistemi server come server per l'elaborazione e file server.

- I **server per l'elaborazione** forniscono un'interfaccia a cui i client possono inviare una richiesta (per esempio, la lettura di alcuni dati); da parte sua, il server esegue l'azione richiesta e restituisce i risultati al client. Un server che ospiti una base di dati a cui i client possono attingere costituisce un esempio di tale sistema.
- I **file server** offrono un'interfaccia al file system che consente al client creazione, aggiornamento, lettura e cancellazione dei file. Un esempio di questo sistema è dato da un server web che trasferisce i file richiestigli dai browser dei client.

1.12.3 Computazione peer-to-peer

Un'altra architettura dei sistemi distribuiti è il modello di sistema da pari a pari, o peer-to-peer (P2P). In tale modello cade la distinzione tra client e server; infatti, tutti i nodi all'interno del sistema sono su un piano di parità, e ciascuno può fungere ora da client, ora da server, a seconda che stia richiedendo o fornendo un servizio. Questi sistemi offrono un vantaggio rispetto a quelli client-server: infatti, alla lunga il server diviene sovraccarico, mentre in un sistema peer-to-peer, uno stesso servizio può essere fornito da uno qualunque dei vari nodi distribuiti nella rete.

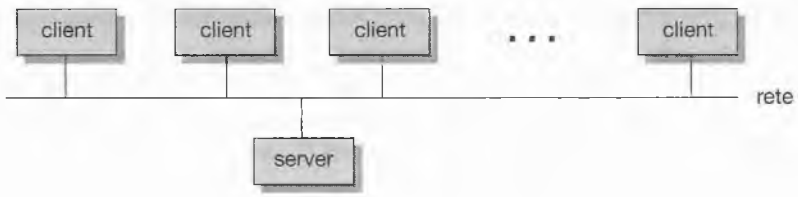

Figura 1.13 Struttura generale di un sistema client-server.

Per entrare a far parte di un sistema peer-to-peer, un nodo deve in primo luogo unirsi ai suoi omologhi che formano la rete. Una volta entrato a far parte della rete, esso può iniziare a fornire i servizi agli altri nodi che risiedono nella rete e, a sua volta, ottenerli dagli altri nodi. Vi sono due modalità generali per stabilire quali servizi siano disponibili.

- Al momento di unirsi a una rete, un nodo iscrive il proprio servizio in un registro centralizzato di consultazione della rete. Quando un nodo vuole ottenere un servizio, esso contatta in via preliminare il registro centralizzato di consultazione, per verificare quali nodi lo forniscano. Il resto della comunicazione ha luogo tra il client e il fornitore del servizio.
- Un risorsa di rete che operi in veste di client deve innanzitutto accertare quale nodo fornisca il servizio desiderato, e lo fa inoltrando la propria richiesta di servizio a tutti gli altri nodi della rete. I nodi che possono fornire tale servizio rispondono al nodo da cui è partita la richiesta. Questa procedura richiede un *protocollo di scoperta*, che deve permettere ai nodi di scoprire i servizi forniti dagli altri nodi della rete.

Le reti paritetiche hanno avuto larga diffusione dalla fine degli anni '90 grazie a diversi servizi di condivisione dei file, quali Napster e Gnutella, che permettono agli utenti lo scambio reciproco di file. Il sistema Napster utilizza una modalità operativa simile al primo tipo descritto precedentemente: un server centrale conserva il registro di tutti i file prelevabili dai nodi della rete Napster, e lo scambio effettivo dei file avviene su iniziativa dei nodi stessi. Il sistema Gnutella adotta una tecnica simile al secondo tipo descritto: un client fa pervenire le richieste di file agli altri nodi del sistema, quindi i nodi che possono soddisfare la richiesta rispondono direttamente al client. In prospettiva futura lo scambio dei file lascia adito a dubbi, visto che molti di loro sono tutelati dalle leggi sui diritti d'autore (i brani musicali, per esempio) che ne regolamentano la distribuzione. In ogni caso, la tecnologia peer-to-peer giocherà, con ogni probabilità, un ruolo determinante nell'ambito di molti servizi, quali la ricerca all'interno di una rete, lo scambio di file e la posta elettronica.

1.12.4 Computazione basata sul Web

La rete Internet è diventata onnipresente, come dimostra il fatto che vi accedono molti più dispositivi di quanti se ne potessero immaginare solo fino a qualche anno fa. I PC, insieme alle stazioni di lavoro, ai PDA e persino ai telefoni cellulari, sono tuttora i dispositivi d'eccellenza per la connessione al Web.

La computazione basata sul Web ha accentuato l'importanza dell'interconnessione in rete. Alcuni dispositivi che prima non avevano la possibilità di collegarsi, garantiscono ora l'accesso, con o senza l'ausilio di cavi. I dispositivi che già godevano della connessione possono ora usufruire di collegamenti di qualità e velocità superiori, che derivano da una migliore tecnologia delle reti, dal perfezionamento dei programmi che gestiscono le reti, o da entrambi i fattori.

La computazione basata sul Web ha dato origine a nuovi dispositivi, quali i **bilanciatori del carico** (*load balancers*), che distribuiscono le connessioni di rete su una serie di server simili. I sistemi operativi come Windows 95, che aveva il ruolo di client web, si sono evoluti in Linux e Windows XP, che possono fungere sia da server web che da client web. Più in generale, il Web ha avuto come effetto collaterale un incremento di complessità dei dispositivi, che devono adesso garantire la possibilità di operare sul Web.

1.13 Sistemi operativi open-source

Lo studio dei sistemi operativi, come già sottolineato, è semplificato dalla disponibilità di un vasto numero di programmi open-source. I **sistemi operativi open-source** sono disponibili in formato sorgente anziché come codice binario compilato. Linux è il più diffuso sistema operativo open-source, mentre Microsoft Windows è un ben noto esempio dell'approccio opposto, a **sorgente chiuso** (**closed source**). Avere a disposizione il codice sorgente permette al programmatore di produrre il codice binario, eseguibile da un sistema. Il processo inverso, chiamato processo di **reverse-engineering**, che permette di ricavare il codice sorgente partendo dal binario, è molto più oneroso; molti elementi utili, ad esempio i commenti, non possono essere ripristinati. Apprendere il funzionamento dei sistemi operativi esaminandone il codice sorgente originale, piuttosto che leggendo descrizioni di quel codice sorgente, può essere molto utile. Avendo a disposizione il codice sorgente, uno studente può modificare il sistema operativo per poi compilare ed eseguire il codice, verificando così i cambiamenti che vi ha apportato. Procedere in questo modo è sicuramente di notevole aiuto per l'apprendimento. Questo libro include degli esercizi che richiedono la modifica del codice sorgente di un sistema operativo. Alcuni algoritmi, inoltre, sono descritti ad alto livello, per essere sicuri di coprire tutti gli argomenti importanti che riguardano i sistemi operativi. Nel libro si possono inoltre trovare riferimenti ad esempi di codice open-source, per eventuali approfondimenti.

I vantaggi dei sistemi operativi open-source sono molti. Tra questi vi è la presenza di una comunità di programmatori interessati (e spesso non retribuiti) che contribuiscono allo sviluppo aiutando a verificare la presenza di eventuali errori nel codice, ad analizzarlo, a dare assistenza e a suggerire dei cambiamenti. I programmi open-source sono, a ragion veduta, più sicuri di quelli a sorgente chiuso, perché molti più occhi sono puntati sul codice. Certo, anche i codici open-source hanno dei bachi ma, come argomentano i difensori dell'open-source, questi bachi vengono scoperti ed eliminati molto più velocemente proprio grazie al gran numero di utilizzatori. Le società che traggono profitto dalla vendita dei loro programmi sono riluttanti a rendere accessibili i loro sorgenti, anche se le aziende che stanno procedendo in questa direzione, come Red Hat, SUSE, Sun e molte altre, dimostrano di trarne benefici commerciali, anziché soffrirne. Per tali società il profitto deriva, ad esempio, da contratti di assistenza e dalla vendita di hardware sul quale il software funziona.

1.13.1 Storia

Ai primordi dell'informatica moderna (ovvero negli anni '50 del secolo scorso) gran parte del software era disponibile in formato open-source. Gli hacker di allora (gli appassionati dei computer) lasciavano i loro programmi nei cassetti del Tech Model Railroad Club (il club del modellismo ferroviario) del MIT affinché altri potessero lavorarci. Gruppi di utenti "casalinghi" scambiavano il codice durante i loro incontri. Qualche tempo dopo, gruppi di utenti legati a specifiche società, come la Digital Equipment Corporation, accettarono contributi al codice sorgente di programmi e li raccolsero su nastri per poi distribuirli ai membri interessati.

Successivamente le società informatiche cercarono di limitare l'utilizzo del loro software a computer autorizzati e clienti paganti. Riuscirono a raggiungere questo obiettivo rendendo disponibili solo i file in binario compilati a partire dal codice sorgente, ma non il codice stesso. Esse protessero così allo stesso tempo il proprio codice e le proprie idee dai concorrenti. Altra questione è quella che riguarda il materiale protetto da copyright. I sistemi operativi e altri programmi possono limitare la possibilità di accedere a film, musica e a libri

elettronici solo a computer autorizzati. Tale **protezione** o **gestione dei diritti digitali** (*digital rights management*, DRM) non sarebbe efficace se il codice sorgente che implementa questi limiti fosse pubblicato. Le leggi di molti paesi, incluso il Digital Millennium Copyright Act (DMCA) negli Stati Uniti, rendono illegale ricavare un codice DRM tramite il reverse-engineering o provare a eludere la protezione del materiale.

Per contrastare la limitazione nell'utilizzo e nella redistribuzione di software, nel 1983 Richard Stallman diede vita al progetto GNU con la finalità di creare un sistema operativo gratuito, open-source e compatibile con UNIX. Nel 1985 pubblicò il manifesto GNU, sostenendo che tutti i software dovrebbero essere gratuiti e open-source. Costituì inoltre la **Free Software Foundation** (FSF) con lo scopo di incoraggiare il libero scambio dei codici sorgente e il libero utilizzo del software. Anziché depositare il proprio software, la FSF distribuisce il software incoraggiandone la condivisione e il miglioramento. La **General Public License** della GNU (GPL) codifica questa distribuzione ed è un'autorizzazione pubblica per il rilascio di software. Fondamentalmente la GPL richiede che il codice sorgente sia distribuito assieme all'eseguibile binario e che qualsiasi cambiamento apportato a quel sorgente sia anch'esso reso disponibile con la stessa autorizzazione GPL.

1.13.2 Linux

Consideriamo **GNU/Linux** come esempio di sistema operativo open-source. Il progetto GNU produsse molti strumenti compatibili con UNIX, inclusi compilatori, editor, utilità, ma non ha mai distribuito un kernel. Nel 1991 uno studente finlandese, Linus Torvalds, rilasciò un kernel rudimentale simile a UNIX utilizzando compilatori e strumenti di GNU e invitò gli interessati a contribuire allo sviluppo. Con l'avvento di Internet, chiunque fosse interessato al progetto poteva scaricare il codice sorgente, modificarlo e sottoporre i cambiamenti a Torvalds. Il rilascio settimanale di aggiornamenti permise a questo sistema operativo, il cosiddetto Linux, di crescere rapidamente, avvalendosi delle migliorie apportate da migliaia di programmatori.

Il sistema operativo GNU/Linux che ne è risultato ha creato centinaia di singole **distribuzioni** del sistema, ovvero versioni personalizzate. Le principali distribuzioni includono Red Hat, SUSE, Fedora, Debian, Slackware e Ubuntu. Le distribuzioni differiscono nelle funzionalità, nelle applicazioni installate, nel supporto hardware, nell'interfaccia e negli obiettivi. Ad esempio, Red Hat Enterprise Linux è indirizzato al grande uso commerciale. PCLinuxOS è un **LiveCD**, un sistema operativo che può essere avviato ed eseguito da un CD-ROM senza essere installato sul disco fisso. Una variante di PCLinuxOS, "PCLinuxOS Supergamer DVD", è un **LiveDVD** che include driver per la grafica e giochi. Un giocatore può farlo funzionare su qualsiasi sistema compatibile semplicemente avviandolo dal DVD; al termine del gioco il riavvio del sistema ripristina il sistema operativo originario.

L'accesso al codice sorgente di Linux varia secondo la versione. Qui prendiamo in considerazione Ubuntu, una distribuzione di Linux disponibile in molte tipologie, comprese quelle destinate ai desktop, ai server o agli studenti. Il suo fondatore si fa carico delle spese per la stampa e la spedizione dei DVD con il codice binario e sorgente (il che contribuisce a diffondere la distribuzione). I seguenti passi permettono di esplorare il codice sorgente del kernel Ubuntu su sistemi che supportano lo strumento gratuito "WMware Player".

Scaricate il software di virtualizzazione (player) da:

- http://www.wmware.com/download/player/ e installatelo sul vostro sistema.
- Scaricate una macchina virtuale contenente Ubuntu. Centinaia di immagini di macchine virtuali (*appliances*), preinstallate con sistemi operativi e applicazioni, sono disponibili su WMware all'indirizzo http://www.wmware.com/appliances/.

- Avviate la macchina virtuale in ambiente WMware player.
- Ottenete il codice sorgente della versione del kernel che vi interessa, ad esempio 2.6, eseguendo `wget` http://www.kernel.org/pub/linux/kernel/v2.6/linux.2.6.18.1.tar.bz2.
- Decomprimete il file scaricato con `tar xjf linux-2.6.18.1.tar.bz2`.
- Esplorate il codice sorgente del kernel Ubuntu, che si trova ora in `./linux-2.6.18.1`.

Per ulteriori informazioni su Linux si veda il Capitolo 21. Per quanto riguarda le macchine virtuali, si veda il Paragrafo 2.8.

1.13.3 UNIX BSD

Rispetto a Linux UNIX BSD ha una storia più lunga e complicata. La sua creazione, derivata dallo UNIX di AT&T, risale al 1978 e le sue prime versioni vennero distribuite dalla Università della California a Berkeley (UCB) in codice sorgente e in formato binario, ma non erano open-source, perché era necessaria una licenza della AT&T. Lo sviluppo di UNIX BSD venne rallentato nei successivi anni da una querela della AT&T, ma alla fine una versione completa e open-source del sistema, la 4.4BSD-lite, venne rilasciata nel 1994.

Esattamente come nel caso di Linux, ci sono diverse distribuzioni di UNIX BSD, tra le quali FreeBSD, NetBSD, OpenBSD e DragonflyBSD. Per esaminare nel dettaglio il codice sorgente di FreeBSD è sufficiente scaricare l'immagine della versione desiderata e caricarla in VMware, come descritto in precedenza per Linux Ubuntu. Il codice sorgente è allegato alla distribuzione e lo si può trovare in `/usr/src`. Il codice sorgente del kernel si trova in `/usr/src/sys`. Per esaminare, ad esempio, il codice che implementa la memoria virtuale nel kernel di FreeBSD, è sufficiente andare in `/usr/src/sys/vm`.

Darwin, il cuore del kernel di MAC, è basato su UNIX BSD ed è anch'esso open-source. Il sorgente è disponibile all'indirizzo http://www.opensource.apple.com/darwinsource/. Lo stesso sito contiene tutte le componenti open-source delle distribuzioni di MAC. Il nome del pacchetto contenente il kernel è "xnu". Il codice sorgente del kernel di MAC nella sua revisione 1228 (il kernel di Leopard MAC) si può trovare all'indirizzo http://www.opensource.apple.com/darwinsource/tarballs/apsl/xnu-1228.tar.gz. Apple fornisce anche diversi strumenti per sviluppatori, documentazione e supporto all'indirizzo http://connect.apple.com.

1.13.4 Solaris

Solaris è il sistema operativo commerciale basato su UNIX della Sun Microsystems. In origine il sistema operativo della Sun, il SunOS, si basava su UNIX BSD. A partire dal 1991Sun iniziò a utilizzare come base del suo sistema operativo UNIX System V di AT&T. Nel 2005 Sun rese disponibile parte del sistema Solaris; col tempo la società ha continuato a fare aggiunte al codice open-source. Purtroppo Solaris non è completamente open-source perché parte del codice è ancora di proprietà della AT&T e di altre società. Tuttavia Solaris può essere compilato dal codice sorgente e collegato (tramite il linker) al codice binario delle componenti a sorgente chiuso. Anche Solaris, quindi, può essere esplorato, modificato, compilato e testato.

Il codice sorgente è disponibile all'indirizzo http://opensolaris.org/os/downloads/. Sono disponibili anche distribuzioni precompilate basate sul codice sorgente, nonché gruppi di discussione. Non è necessario scaricare l'intero pacchetto del codice sorgente dal sito, perché Sun permette ai visitatori di esplorare il codice sorgente online con un apposito browser.

1.13.5 Utility

Il movimento a sostegno dei software gratuiti spinge legioni di programmatori a creare migliaia di progetti open-source, inclusi sistemi operativi. Siti come http://freshmeat.net/ e http://distrowatch.com/ offrono portali per questi progetti. I progetti open-source permettono agli studenti di utilizzare il codice sorgente come strumento di apprendimento, dando loro l'opportunità di modificare i programmi, testarli, contribuire all'individuazione di bachi e alla loro eliminazione, oltre a esplorare sistemi operativi maturi e completamente costruiti, con i relativi compilatori, strumenti, interfaccia e altri programmi. La disponibilità del codice sorgente per progetti storici, come Multics, può aiutare gli studenti a comprendere quei progetti e a costruire conoscenze utili.

GNU/Linux, UNIX BSD e Solaris sono tutti sistemi operativi open-source, ma ognuno ha obiettivi, funzionalità, licenze e scopi propri. A volte le licenze non si escludono reciprocamente e danno vita a una sorta di impollinazione incrociata che contribuisce a un rapido miglioramento dei progetti riguardanti i sistemi operativi. Ad esempio, diverse componenti di rilievo di Solaris sono state trasferite su UNIX BSD. I vantaggi del software gratuito e dell'open-source possono portare a un aumento del numero e della qualità dei progetti open-source, comportando un incremento del numero di individui e società che li utilizzano.

1.14 Sommario

Un sistema operativo è il software che gestisce l'hardware di un calcolatore, e fornisce un ambiente all'interno del quale siano eseguibili le applicazioni. Forse l'aspetto più concreto dei sistemi operativi è l'interfaccia d'accesso al computer che essi forniscono all'utente.

Per essere eseguiti, i programmi devono risiedere nella memoria centrale del calcolatore. Questa è infatti la sola area di memoria di grandi dimensioni direttamente accessibile dalla CPU. Essa è un vettore, di parole o byte, di dimensioni variabili tra i milioni e i miliardi di byte. Ciascuna parola possiede il proprio indirizzo. La memoria centrale è un dispositivo volatile, poiché perde il proprio contenuto quando manca l'alimentazione elettrica. La maggior parte dei sistemi di calcolo dispone di una memoria secondaria come estensione della memoria centrale. Il requisito fondamentale della memoria secondaria è la capacità di memorizzare in modo permanente grandi quantità di dati. Il più comune dispositivo di memoria secondaria è il disco magnetico; si tratta di un dispositivo di memoria non volatile che può memorizzare sia dati sia programmi e che consente l'accesso diretto ai suoi elementi.

I sistemi di memorizzazione di un calcolatore si possono organizzare in modo gerarchico secondo la velocità e il costo. I livelli più alti rappresentano i dispositivi più rapidi, ma più costosi. Scendendo nella gerarchia, il costo per bit generalmente decresce, mentre di solito aumentano i tempi d'accesso.

Alla base della progettazione di un sistema di elaborazione sono possibili approcci diversi. Una prima scelta deve essere operata tra i sistemi a processore unico e quelli con due o più processori; in quest'ultimo caso, la memoria fisica e i dispositivi periferici sono condivisi da tutti i processori. Lo schema più comune tra i sistemi multiprocessore è rappresentato dalla multielaborazione simmetrica (SMP), che vede tutti i processori su un piano di parità: essi elaborano in piena indipendenza gli uni dagli altri. Una forma specializzata di multielaborazione è invece rappresentata dai cluster di elaboratori (*clustered systems*), ovvero un certo numero di elaboratori connessi da una rete locale.

Per utilizzare al meglio la CPU, i sistemi operativi moderni impiegano la multiprogrammazione, grazie alla quale diversi processi possono occupare contemporaneamente la memoria, assicurando che la CPU non resti mai inattiva. Con i sistemi a tempo ripartito il concetto di multiprogrammazione è stato ulteriormente esteso, per mezzo di algoritmi di scheduling della CPU che fanno la spola tra un processo e l'altro, dando così l'impressione che molti processi siano in esecuzione allo stesso tempo.

Il sistema operativo deve assicurare il corretto funzionamento del calcolatore. Per evitare che i programmi utenti interferiscano tra di loro e col sistema operativo, la CPU ha due modalità di funzionamento: la modalità utente e la modalità di sistema. Diverse istruzioni (come quelle di I/O e di arresto del calcolatore) sono privilegiate e si possono eseguire solamente in modalità di sistema. Anche l'area della memoria in cui risiede il sistema operativo deve essere protetta dai tentativi di modifiche da parte degli utenti. La presenza di un timer evita il verificarsi di cicli infiniti. Queste funzioni (duplice modalità di funzionamento, istruzioni privilegiate, protezione della memoria, interruzioni del timer) costituiscono gli elementi fondamentali impiegati dal sistema operativo per ottenere un corretto funzionamento del sistema.

Un processo (*process* o *job*) è l'unità fondamentale di lavoro in un sistema operativo. La gestione dei processi comprende aspetti come la loro creazione e cancellazione, nonché la messa a punto di meccanismi per la comunicazione reciproca e la sincronizzazione dei processi. Un sistema operativo, nel gestire la memoria, si cura di registrare quali parti di essa vengano usate e da chi. È sempre al sistema operativo, inoltre, che spetta l'allocazione dinamica e il rilascio dello spazio di memoria. Esso gestisce anche l'archiviazione dei dati: dai file system per i file e le directory, ai dispositivi per la memorizzazione di massa.

Altre due indispensabili funzioni dei sistemi operativi sono le strategie di protezione e le politiche per la sicurezza del sistema. La protezione è garantita da una serie di meccanismi per il controllo dell'accesso, da parte dei processi o degli utenti, alle risorse che il sistema mette a disposizione. Alle misure di sicurezza è invece affidata la difesa dell'elaboratore da attacchi esterni o interni.

I sistemi distribuiti conferiscono agli utenti la possibilità di condividere risorse che giacciono a grande distanza le une dalle altre, su una rete di terminali connessi. I servizi possono essere offerti sia con il modello client-server che con il modello peer-to-peer (ossia, da pari a pari). In un cluster, poiché i dati risiedono in una memoria condivisa, l'elaborazione può essere compiuta da più macchine; di conseguenza, anche se alcuni membri del cluster dovessero subire dei guasti, l'elaborazione può essere portata avanti dagli altri.

I due tipi fondamentali di reti sono le LAN e le WAN. Le prime consentono l'interazione tra processori collocati in un raggio geografico ristretto, mentre le seconde mettono in comunicazione processori separati da distanze più grandi. Le LAN sono generalmente più veloci delle WAN.

Vi sono diversi sistemi di calcolo che perseguono scopi specifici. Ne fanno parte i sistemi operativi real-time destinati ad ambienti integrati quali i prodotti commerciali, le automobili e la robotica. I sistemi operativi real-time sono vincolati da scadenze temporali ben definite, che non tollerano ritardi. L'elaborazione deve essere completata entro i limiti prestabiliti, pena il fallimento del sistema. I sistemi multimediali riguardano la trasmissione di dati multimediali, e hanno spesso requisiti particolari per la visualizzazione o la riproduzione di filmati e tracce audio, o casi in cui audio e video sono sincronizzati.

In tempi recenti, l'influenza di Internet e del Web ha incoraggiato lo sviluppo di sistemi operativi moderni che includono come parti integranti elementi quali i browser web, il software per l'accesso alla rete e i programmi per la comunicazione in rete.

Il movimento a sostegno del software gratuito ha creato migliaia di progetti open-source, inclusi sistemi operativi. Grazie a questi progetti gli studenti hanno la possibilità di utilizzare il codice sorgente come strumento di apprendimento. Possono infatti modificare i programmi e testarli, individuare bachi ed eliminarli, e al tempo stesso esplorare sistemi operativi maturi e completamente definiti, compilatori, strumenti, interfacce e altri tipi di programmi.

GNU/Linux, UNIX BSD e Solaris sono tutti sistemi operativi open-source, ma ognuno ha obiettivi, utilità, licenze e scopi propri. A volte le licenze non si escludono reciprocamente e danno vita a una sorta di impollinazione incrociata che contribuisce a un rapido miglioramento dei progetti riguardanti i sistemi operativi. Ad esempio, diverse componenti di rilievo di Solaris sono state trasferite su UNIX BSD. I vantaggi del software gratuito e dell'open-source possono portare a un aumento del numero e della qualità dei progetti open-source, comportando un incremento del numero di individui e società che li utilizzano.

Esercizi pratici

1.1 Quali sono i tre scopi principali di un sistema operativo?

1.2 Quali sono le differenze principali tra i sistemi operativi per mainframe e quelli per personal computer?

1.3 Elencate le quattro fasi necessarie per eseguire un programma su una macchina completamente dedicata, ovvero un computer sul quale viene eseguito solo quel programma.

1.4 Abbiamo sottolineato come il sistema operativo sia volto all'uso efficiente dell'hardware. Quando è opportuno che il sistema operativo rinunci a questo principio e "sprechi" risorse? Perché un sistema simile non può essere considerato davvero inefficiente?

1.5 Qual è la difficoltà principale che deve superare un programmatore nello scrivere un sistema operativo per un ambiente real-time?

1.6 Considerate le varie definizioni di sistema operativo. Valutate se sia opportuno che il sistema operativo includa o meno applicazioni quali browser e programmi di posta elettronica. Argomentate entrambe le possibilità, fornendo delle motivazioni.

1.7 Come funziona la distinzione tra modalità di sistema (modalità kernel) e modalità utente quale rudimentale forma di protezione (sicurezza) del sistema?

1.8 Quale delle seguenti istruzioni dovrebbe essere privilegiata?

a. Impostare il timer.
b. Leggere il clock.
c. Cancellare la memoria.
d. Invocare un'istruzione trap.
e. Disattivare le interruzioni.
f. Modificare le informazioni nella tabella che indica lo status dei dispositivi.
g. Passare da modalità di sistema a modalità utente.
h. Accedere a un dispositivo I/O.

1.9 Alcuni dei primi computer proteggevano il sistema operativo posizionandolo in una partizione della memoria che non poteva essere modificata né dall'utente né dal sistema operativo. Descrivete due difficoltà che secondo voi potrebbero sorgere da uno schema simile.

1.10 Alcune CPU offrono più di due modalità di operazione. Quali sono due possibili impieghi di queste modalità multiple?

1.11 I timer potrebbero essere utilizzati anche per calcolare l'ora corrente. Spiegate brevemente come.

1.12 Internet è una rete LAN o WAN?

Esercizi

1.13 In un ambiente multiprogrammato e a tempo ripartito, diversi utenti condividono il sistema simultaneamente. Tale situazione può generare alcuni problemi di sicurezza.

a. Individuate due possibili problemi.
b. Si può garantire il medesimo grado di sicurezza in una macchina a tempo ripartito e in una macchina dedicata? Motivate la risposta.

1.14 La tematica dell'utilizzo delle risorse è una costante dei sistemi operativi, seppure in modi diversi a seconda del tipo di sistema considerato. Si dettaglino le risorse da gestire con cura nelle seguenti situazioni:

a. mainframe o minicomputer;
b. stazioni di lavoro connesse a server;
c. computer palmari.

1.15 Quando e perché sarebbe più conveniente per un utente scegliere un sistema a tempo ripartito anziché un PC o una stazione di lavoro individuale?

1.16 Quale tra le funzionalità citate in basso deve possedere un sistema operativo in queste situazioni: (a) dispositivi palmari e (b) sistemi real-time.

a. Programmazione a lotti.
b. Memoria virtuale.
c. Tempo ripartito.

1.17 Descrivete le differenze tra multielaborazione simmetrica e asimmetrica. Elencate tre vantaggi e uno svantaggio dei sistemi multiprocessore.

1.18 Quali differenze presentano i cluster di elaboratori rispetto ai sistemi per la multielaborazione? Che cosa è necessario perché due macchine appartenenti a un cluster, con il loro contributo congiunto, offrano un servizio altamente affidabile?

1.19 Tratteggiate le differenze tra il modello client-server e quello peer-to-peer.

1.20 Ipotizziamo che sui due nodi di un cluster di elaboratori sia attiva una base di dati. Descrivete due modalità con cui il software per la gestione del cluster può regolare l'accesso ai dati sul disco, analizzando i pro e i contro di ognuna.

1.21 In che cosa i computer di rete si differenziano dagli elaboratori tradizionali? Enunciate qualche caso concreto in cui sia preferibile utilizzare un computer di rete.

1.22 Qual è lo scopo delle interruzioni? Quali differenze vi sono tra un'eccezione e un'interruzione? Un programma utente può generare un'eccezione di proposito? In caso affermativo, con quale scopo?

1.23 L'accesso diretto alla memoria (DMA) è usato per dispositivi I/O ad alta velocità per evitare di sovraccaricare la CPU.

a. In che modo la CPU si interfaccia con il dispositivo per coordinare il trasferimento?
b. In che modo la CPU apprende che il trasferimento in memoria è completo?
c. La CPU è abilitata all'esecuzione di altri programmi mentre il controllore DMA procede al trasferimento dei dati. Può tale trasferimento interferire con la corretta esecuzione dei programmi utenti? Se la risposta è positiva, descrivete in quale forma può sorgere l'interferenza.

1.24 L'architettura di alcuni sistemi di calcolo non possiede una modalità di funzionamento riservata al sistema operativo. Spiegate se per questi calcolatori è possibile realizzare sistemi operativi sicuri.

1.25 Fornite due motivi che dimostrino l'utilità delle cache, quali problemi risolvono e quali provocano. Potendo disporre di una cache delle dimensioni del dispositivo cui è associata (per esempio una cache delle dimensioni di un disco) spiegate se sia possibile usarla eliminando il dispositivo.

1.26 Considerate un sistema SMP simile a quello della Figura 1.6. Illustrate con un esempio come i dati presenti nella memoria potrebbero avere due valori differenti in ognuna delle cache locali.

1.27 Illustrate, con l'ausilio di esempi, come si manifesta il problema della coerenza dei dati memorizzati nella cache nei seguenti ambienti di elaborazione:

a. sistemi a processore unico;
b. sistemi multiprocessore;
c. sistemi distribuiti.

1.28 Descrivete un meccanismo di protezione della memoria grazie al quale sia possibile impedire a un programma la modifica della memoria di pertinenza di altri programmi.

1.29 Dite quale configurazione di rete sia più adatta ai seguenti ambienti:

a. un piano di un pensionato universitario;
b. un campus universitario;
c. una regione;
d. una nazione.

1.30 Si definiscano le caratteristiche tipiche appartenenti a ciascun sistema operativo:

a. a lotti;
b. interattivo;
c. a tempo ripartito;
d. in tempo reale;
e. di rete;
f. parallelo;

g. distribuito;
h. cluster;
i. palmare.

1.31 Quali sono i vantaggi e gli svantaggi offerti dai computer palmari?

1.32 Identificate vantaggi e svantaggi dei sistemi operativi open-source. Discutete anche le tipologie di persone che potrebbero definire determinati aspetti come vantaggi oppure svantaggi.

1.15 Note bibliografiche

[Brookshear 2003] fornisce una panoramica a grandi linee sull'informatica in generale. [Bovet e Cesati 2002] presentano la struttura di base del sistema operativo Linux. L'opera di [Solomon e Russinovich 2000], oltre a dare un'idea generale su Windows, è ricca di dettagli tecnici sui componenti del sistema. Russinovich e Salomon [2005] hanno aggiornato questa informazione fino a Windows Server 2003 e Windows XP. McDougall e Mauro [2007] trattano dei componenti del sistema operativo Solaris. Mac OS X è presente su http://www.apple.com/macosx. I componenti di Mac OS X vengono trattati in Singh [2007].

Tra quanti hanno esplorato i sistemi peer-to-peer citiamo [Parameswaran et al. 2001], [Gong 2002], [Ripeanu et al. 2002], [Agre 2003], [Balakrishnan et al. 2003] e [Loo 2003]. In [Lee 2003] è reperibile una discussione sui sistemi peer-to-peer per la condivisione dei file. Un'opera di riferimento per i cluster di elaboratori è quella di [Buyya 1999]. Sviluppi più recenti dello stesso argomento sono descritti da [Ahmed 2000]. Una rassegna di questioni relative ai sistemi distribuiti è fornita da [Tanenbaum e Van Renesse 1985].

I sistemi operativi trovano spazio in numerosi libri di testo generali, tra cui [Stallings 2000b], [Nutt 2004] e [Tanenbaum 2001].

[Hamacher et al. 2002] descrivono l'organizzazione degli elaboratori. McDougall e Laudon [2006] discutono dei processori multicore. Hennessy e Patterson [2007] trattano dei sistemi di I/O, dei canali di comunicazione e delle architetture di sistema in genere. Blaauw e Brooks [1997] descrivono dettagliatamente l'architettura di molti sistemi di elaborazione, compresi molti sistemi IBM. Stokes [2007] fornisce un'introduzione illustrata ai microprocessori e all'architettura dei calcolatori.

La memoria cache, compresa la memoria associativa, è analizzata in [Smith 1982], un articolo che offre una vasta bibliografia sull'argomento.

Alcuni approfondimenti riguardanti la tecnologia dei dischi magnetici si trovano in [Freedman 1983] e [Harker et al. 1981], mentre i dischi ottici sono trattati da [Kenville 1982], [Fujitani 1984], [O'Leary e Kitts 1985], [Gait 1988] e [Olsen e Kenley 1989]. Sui floppy disk si soffermano [Pechura e Schoeffler 1983] e anche [Sarisky 1983]. Considerazioni generali sulle tecnologie per l'archiviazione di massa dei dati sono offerte da [Chi 1982] e [Hoagland 1985].

[Kurose e Ross 2005] e [Tanenbaum 2003] offrono una trattazione generale delle reti di calcolatori. [Fortier 1989] presenta un dettagliato approfondimento del software e dell'hardware impiegato nelle reti. Kozierok [2005] discute il TCP in dettaglio. Mullender [1993] fa una panoramica dei sistemi distribuiti. [Wolf 2003] espone gli sviluppi recenti relativi ai sistemi integrati. Questioni legate ai dispositivi palmari sono illustrate da [Myers e Beigl 2003], [Di Pietro e Mancini 2003].

Una trattazione completa della storia dell'open-source e dei suoi vantaggi e sfide si può trovare in Raymond [1999]. La storia della pirateria informatica (*hacking*) è discussa da Levy [1994]. La Free Software Foundation ha pubblicato il suo manifesto all'indirizzo http://gnu.org/philosophy/free-

software-for-freedom.html. Istruzioni dettagliate su come compilare il kernel Linux Ubuntu si trovano su: http://howtoforge.com/kernel_compilation_ubuntu. I componenti open-source di MAC sono disponibili su http://developer.apple.com/open-source/index.html.

Wikipedia (http://en.wikipedia.org/wiki/richard_stallman) contiene una voce su Richard Stallman.

Il codice sorgente di Multics è disponibile su http://web.mit.edu/multics-history/source/multics_internet_server/multics_sources.html.