1 Introduzione ai Sistemi Operativi

Cos'è un Sistema Operativo?

Nella seconda parte di questo volume trattiamo del software che rende possibile il funzionamento di ogni computer: il Sistema Operativo.
Le funzioni di un Sistema Operativo (Operating System, O.S.) sono così numerose e differenti che è difficile darne una definizione meno generale della seguente:

Un **Sistema Operativo** (S.O.) è quell'insieme di programmi che permette ad ogni computer di eseguire il software applicativo.

L'importanza del S.O. è rimarcata dal fatto che esso "modifica" le caratteristiche dell'hardware.
Sullo stesso tipo di hardware possono girare Sistemi Operativi diversi; dunque il S.O. è in grado ci conferire "personalità" diverse a computer che hanno hardware identico. Si pensi per esempio a quanto diverso può essere lo stesso PC se vi si installa il S.O. MS-DOS oppure Windows, Linux o QNX.
Ciò che l'utente comune usa ogni giorno non è dunque il solo hardware del computer, ma l'insieme di hardware e Sistema Operativo.
Chiameremo "**programma di sistema**" un programma che fa parte del Sistema Operativo; chiameremo "**programma d'utente**" o "**applicativo**" un programma che non fa parte del Sistema Operativo.

Cosa NON è un Sistema Operativo

Sulle funzioni del S.O. spesso non c'è molta chiarezza per cui è bene spiegare da subito alcuni tipici fraintendimenti.
Un Sistema Operativo non è:

- un linguaggio di programmazione
  Un linguaggio di programmazione è la specifica astratta di alcune regole che permettono di trasmettere istruzioni ad un computer; un linguaggio di programmazione non è un Sistema Operativo, anzi, non è neppure un programma.
- un compilatore
  Il compilatore è un programma, non diverso da un normale applicativo, che traduce la forma con cui sono scritti altri programmi. Dato che la gran parte degli utenti può benissimo farne a meno, i compilatori non fanno parte del S.O. .
- una libreria di comandi
  Ogni S.O. ha una nutrita libreria di comandi "esterni" (system programs), che hanno caratteristiche simili ai normali programmi dell'utente e possono da questi essere sostituiti (si pensi per esempio a FORMAT.COM o CHKDSK.EXE di MS-DOS).
  Peraltro il S.O. non è composto solamente di questi "comandi" esterni, ma ha anche altri componenti, molto più importanti, che ne costituiscono il "cuore".
  Dunque i comandi di sistema, per quanto si possano considerare parte del S.O. , non "sono" il S.O. .
- un programma che fa eseguire i comandi dell'utente
  Ogni S.O. deve dispone di un programma attraverso il quale l'utente può comunicare quali comandi (programmi) vuole che siano eseguiti. Questo programma prende il nome di "**interprete dei comandi**" e fa parte del Sistema Operativo.
  L'interprete dei comandi è l'"interfaccia" più semplice fra l'utente ed il S.O. Peraltro un Sistema Operativo S.O. non è costituito solamente da un interprete di comandi ma anche da moltissimi altri programmi che fanno un lavoro molto meno "visibile".

1.1 Classificazione dei Sistemi Operativi

Effettuiamo ora una serie di "classificazioni" distinguendo i S.O. in base ad alcune loro importanti caratteristiche.
Prima però è necessario definire un termine che ricorrerà spesso nel seguito. Consideriamo una "**risorsa**" una qualsiasi entità della quale un programma può aver bisogno, come per esempio la memoria centrale (RAM), la memoria di massa (hard disk, CD, nastri, file, record di file) o tutte le periferiche (stampanti, tastiera, scanner ..).
Quello di risorsa è dunque un concetto generico, che nel caso dei S.O. preciseremo meglio in seguito (vedi Gestione delle risorse del sistema)
Le principali differenze fra i diversi Sistemi Operativi si osservano considerando il numero dei programmi che possono essere eseguiti contemporaneamente, e la loro modalità di esecuzione.

S.O. monoprogrammati o multiprogrammati (multitasking)

Un Sistema Operativo nel quale possa girare un solo programma alla volta si dice "**monoprogrammato**" (in Inglese "monoprogrammed o "**single tasking**").

Per esempio MS-DOS è un S.O. monoprogrammato, progettato per l'esecuzione di un solo programma per 8086 alla volta. Quando lanciamo un programma MS-DOS dobbiamo attendere la sua conclusione prima di eseguirne un altro.
Invece un Sistema Operativo è "multiprogrammato" (**multitasking**) quando fa eseguire più di un programma "contemporaneamente". L'uso delle virgolette è d'obbligo ed indica che la contemporaneità nell'esecuzione dei programmi può essere del tutto "simulata".
Sappiamo infatti che in un computer la CPU è l'unico dispositivo che può eseguire un programma; per cui se in un computer esiste una sola CPU, come succede nella maggior parte dei casi, essa può eseguire in ogni istante le istruzioni di un solo programma.
Però, se il S.O. lo permette, è possibile fare in modo che la CPU venga assegnata "a turno" ai diversi programmi che ne hanno bisogno. In questo modo ciascuno di quei programmi potrà evolvere indipendentemente dagli altri.
Dato che l'avvicendamento fra i vari programmi avviene molto rapidamente, per un utente umano "esterno" sarà come se i programmi eseguissero "contemporaneamente".

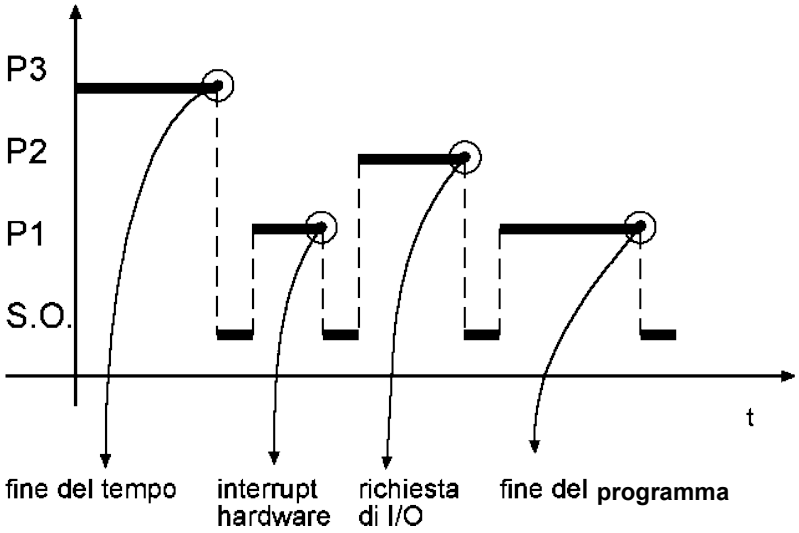


Figura 1: multitasking

Questo ci permette di precisare la definizione di S.O. multitasking, affermando:

Un sistema è multiprogrammato quando più programmi, avviati e non conclusi, esistono contemporaneamente in memoria e progrediscono nella loro esecuzione.

In un sistema multitasking più di un programma è pronto ad eseguire in memoria, in attesa del suo turno per l'utilizzazione della CPU.
Dato che, in un sistema che ha una sola CPU, un solo programma alla volta può eseguire, ne consegue che tutti i programmi sono "sospesi", eccetto quello che sta eseguendo in ogni determinato istante.

Esempio: le versioni attuali dei S.O. Windows e Linux, scritti per CPU di tipo 386 o posteriore, sono multiprogrammate.

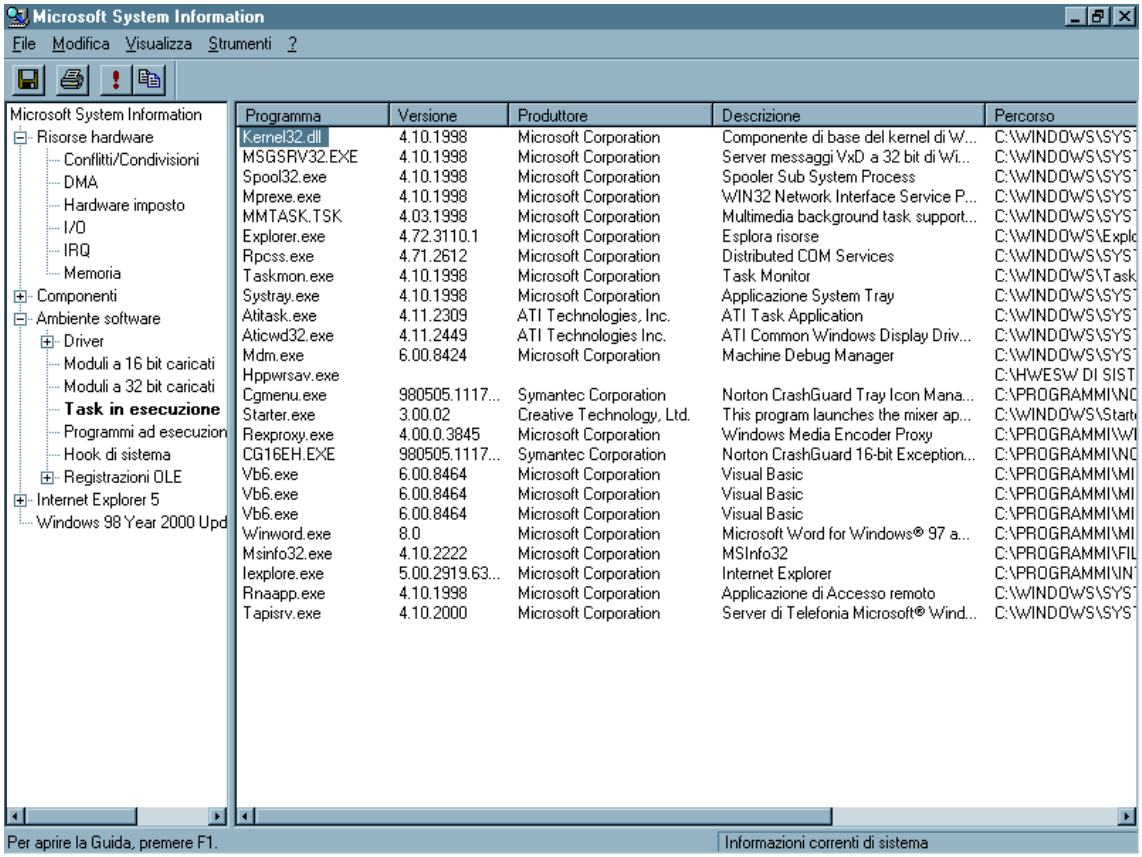


Figura 2: lista dei programmi (processi) in esecuzione contemporaneamente in Windows 98 (sysinfo.exe)

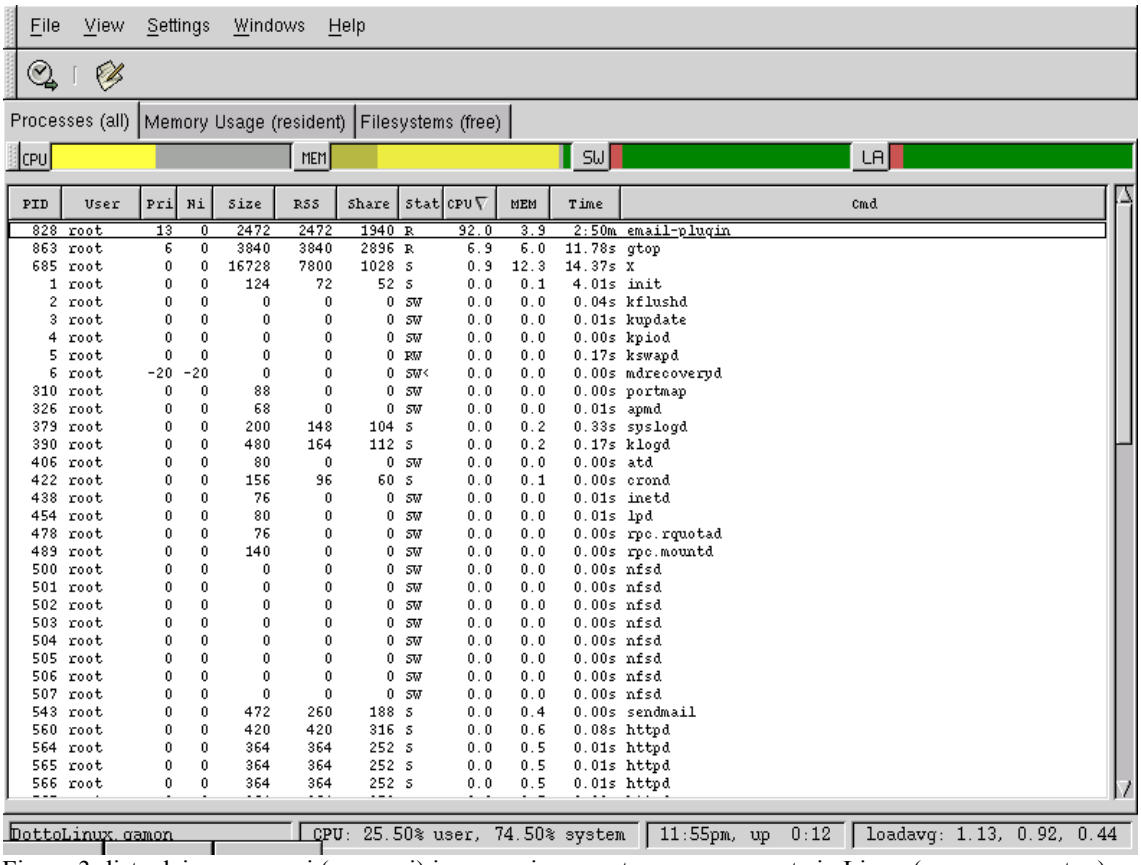


Figura 3: lista dei programmi (processi) in esecuzione contemporaneamente in Linux (programma gtop)

Se eseguiamo il seguente programma di sistema in MS-DOS:
C:> find "stelle" inferno.txt > stellinf.txt

Otteniamo, nel file stelle.txt, la lista delle linee del file inferno.txt che contengono la stringa "stelle". Quando il programma find ha finito la sua esecuzione possiamo lanciarne un altro, al "prompt" del DOS:
C:> find "stelle" purgator.txt > stellprg.txt

Se invece eseguiamo il seguente comando di sistema su un S.O. Unix (p.es. Linux):
grep 'stelle' inferno.txt > stelleInf.txt &
otteniamo lo stesso risultato, ma il programma viene lanciato contemporaneamente ad altri, si torna al "prompt" e se ne può lanciare subito un altro:
grep 'stelle' purgatorio.txt > stellePurg.txt &
Il programma "grep" lanciato per primo esegue contemporaneamente a quello lanciato per secondo ed essi non si disturbano uno con l'altro.

S.O. monoutente o multiutente (multiuser)

Un Sistema Operativo multitasking fa eseguire più di un programma alla volta, ma non è detto che possa servire più di un utente contemporaneamente.
Un S.O. **monoutente** (**single user**) permette ad un solo utente di eseguire programmi sul computer ove è installato.

Alcuni sistemi multitasking sono monoutente (p.es. Windows 9X). Questi sistemi permettono al loro singolo utente di far eseguire molti programmi simultaneamente, ma non gestiscono più di un utente contemporaneamente.

Un S.O. è **multiutente** (**multiuser**) se può eseguire contemporaneamente programmi di utenti diversi.
I S.O. multiutente devono gestire un elenco di utenti "accreditati", ciascun utente dispone di particolari diritti per l'accesso alle risorse del sistema.

login

L'inizio di una sessione di lavoro da parte di un utente viene chiamato "**login**" [1]

Il processo di chiusura di una sessione di lavoro viene, per contrasto, chiamato "**logout**" o "**logoff**".

Ogni utente ha un suo nome convenzionale (**username**), che deve essere digitato "in chiaro", visibile a tutti, al momento dell'accesso, ed un suo "conto", detto "**account**", nel quale vengono registrate tutte le informazioni che lo riguardano.
Ad un account vengono assegnate una cartella di lavoro per i suoi file ed altre risorse varie del sistema.
Il processo di login prevede l'"autenticazione" (authentication) dell'utente, di solito per mezzo della richiesta di una **password** (parola chiave), da digitarsi dopo aver dato lo username, senza che venga fatta "eco" dei tasti digitati (al posto dei dati digitati si vede sempre lo stesso carattere, in modo che chi vede il monitor del computer non la possa acquisire).
Password
La password deve essere conosciuta solo dall'utente e dal Sistema Operativo, neppure l'Amministratore del sistema ne è a conoscenza. Essa deve essere scelta in modo che sia difficile indovinarla.
Un procedimento efficace può essere il seguente:
1. Scegliere una breve frase che si possa ricordare facilmente (p. es. la settimana enigmistica)
2. Togliere le vocali dalla prima parola, le consonanti dalla seconda, le vocali dalla terza, e così via .. (p. es. leiaangmstc)
3. Aggiungere eventualmente qualche carattere speciale, numeri, o cambiare minuscole in maiuscole (p. es. L5Eiaa#Ngmstc!)
4. Si tenga presente il fatto che chi vuole entrare abusivamente in un sistema di elaborazione spesso possiede strumenti automatici, basati su dizionario, per indovinare la password per tentativi. Per questo non è una buona idea usare parole intere, che possono essere indovinate più facilmente.
5. Si eviti anche di inserire nella password informazioni personali, anche non molto note al pubblico, perché si possono indovinare usando la logica e dati personali acquisibili più o meno facilmente, come: data di nascita, numero di telefono, codice fiscale, data del diploma, nome di parenti o amici ..).
Al giorno d'oggi appaiono sistemi più sofisticati di autenticazione dell'utente, da usarsi in aggiunta od in sostituzione della password, quali il riconoscimento dell'impronta digitale, della voce, o dell'iride dell'occhio (tecniche di "identificazione biometrica").

1 "log", in Inglese, significa "diario" o "giornale di bordo" (di una nave). Il termine log-in viene dal fatto che all'inizio della sessione di lavoro viene creato un "file di log" per l'utente che si collega, nel quale il S.O. tiene traccia delle azioni che l'utente ha fatto e delle risorse che ha utilizzato. Queste informazioni servono per sicurezza e contabilizzazione e possono essere: il nome di ogni programma lanciato dall'utente, la quantità di hard disk usato, il tempo di CPU ed in generale la quantità di risorse del sistema che ha utilizzato. Tenere traccia di queste informazioni può essere utile per far pagare all'utente l'uso del sistema in base alle risorse che ha effettivamente utilizzato.

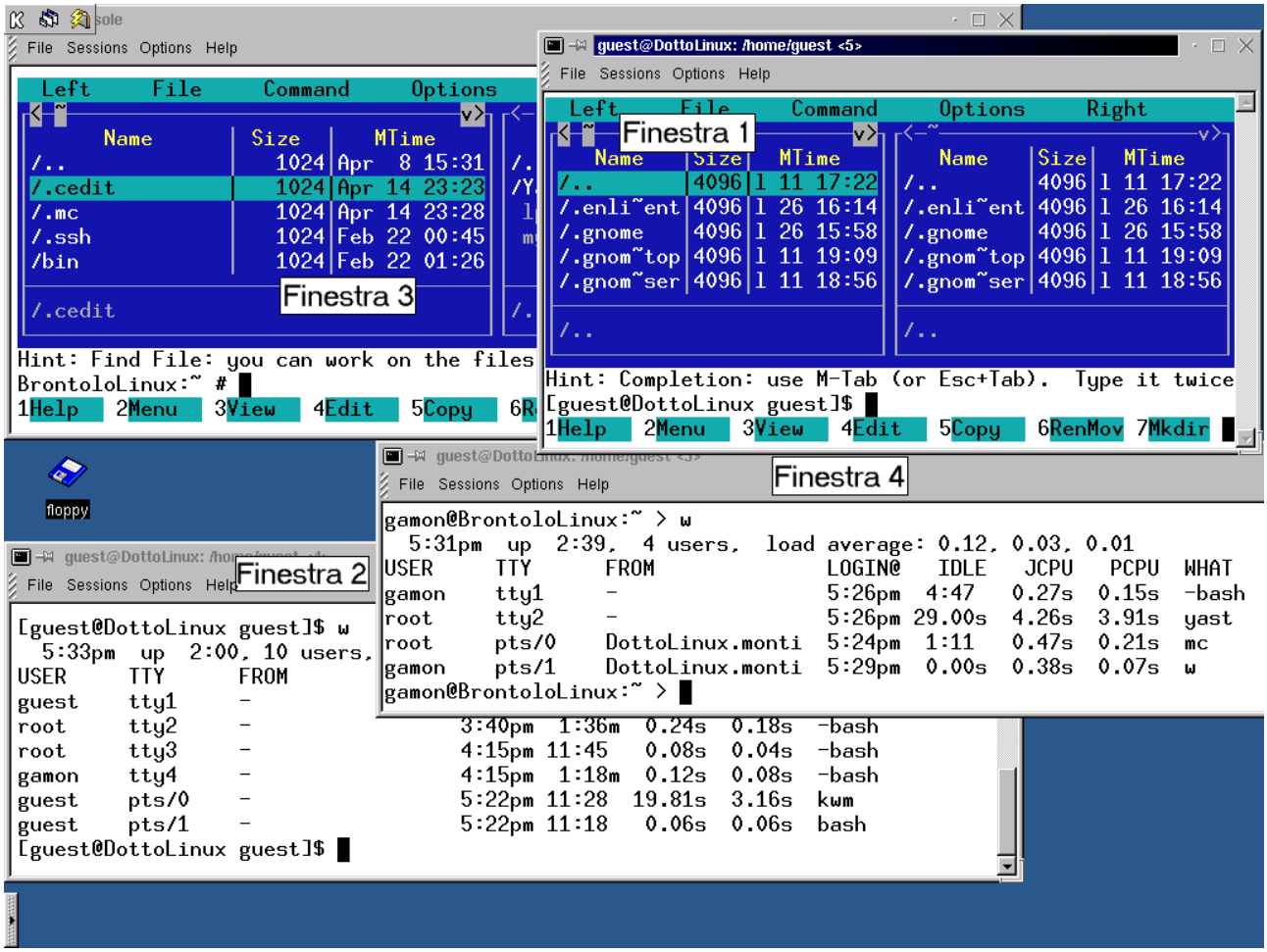


Figura 4: sessioni di lavoro in multiutenza

La Figura 4 mostra 4 finestre grafiche Linux nelle quali diversi utenti si collegano a diversi computer. Nelle finestre 1 e 2 sono mostrate due sessioni delle stesso utente (guest) sul computer DottoLinux, nella finestra 2 sono mostrati tutti gli utenti collegati e tutte le connessioni attive sul computer DottoLinux. Nelle finestre 3 e 4 sono mostrati due collegamenti al computer remoto BrontoloLinux, in finestra 3 è collegato l'utente "root" di BrontoloLinux, in finestra 4 l'utente "gamon". In finestra 3 sono mostrati tutti gli utenti collegati e tutte le connessioni attive sul computer BrontoloLinux.
Il computer su cui gira l'ambiente grafico Linux è DottoLinux.

Amministratore
In ogni sistema multiutente devono esistere un utente od un gruppo utenti "speciali", che hanno particolari diritti sulle risorse del sistema, possono creare gli altri utenti, assegnare ad essi i diritti sulle risorse, stabilire il funzionamento del sistema.
Nei sistemi Unix l'amministratore viene detto "superuser" ed ha il nome d'utente "root", esso è l'unico utente "onnipotente" che esiste nel sistema, ma può cedere alcuni dei suoi diritti ad altri utenti.
Nei sistemi Windows per server di rete (vedi paragrafo successivo) l'amministratore si chiama "Administrator", e se ne può cambiare il nome. Esiste anche un gruppo di utenti "Administrators"; gli utenti che appartengono a questo gruppo hanno gli stessi diritti dell'utente "Administrator".

S.O. monoprocessore o multiprocessore (multiprocessing)

Se in un computer esiste più di una CPU, esso viene detto "**computer multiprocessore**" (multiprocessor computer).
Esistono molti tipi di computer multiprocessore; alcuni contengono solo due CPU altri alcune decine di migliaia.
Un Sistema Operativo che è in grado di utilizzare contemporaneamente le CPU presenti in un computer multiprocessore viene detto S.O. **multiprocessore** (**multiprocessing O.S.**).
Con un Sistema Operativo multiprocessing è possibile una effettiva contemporaneità nell'esecuzione delle istruzioni dei programmi.

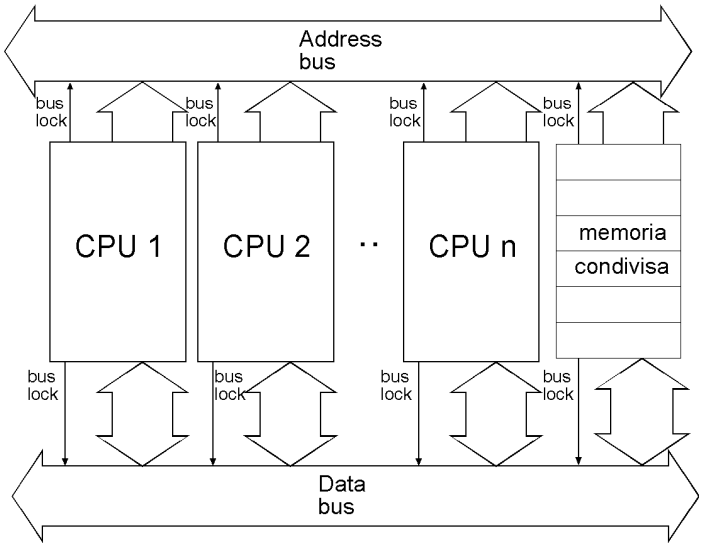


Figura 5: multiprocessing

I S.O. per computer multiprocessore devono essere in grado di suddividere il lavoro fra le diverse CPU presenti. In questo caso la contemporaneità dell'esecuzione delle istruzioni è effettiva, perché ogni CPU presente è in grado di eseguire un programma diverso.

!!!! da fare !
Figura 6: parte della schermata di inizializzazione di due S.O. multiprocessing. (a) Windows (b) Linux

S.O. per server di rete

Un **server** è un dispositivo e/o un programma che rende disponibile una risorsa a diversi utenti o computer ad esso collegati in rete.

Gli "utenti" di un server vengono chiamati **client** e di solito sono collegati al server attraverso una rete di computer (LAN, Internet) .
Un computer può contenere più di un server, ciascuno dedicato ad un risorsa (file server, print server, graphic server, database server, ..).
Una **rete di computer** è un insieme di computer, ciascuno in grado di funzionare autonomamente, in grado di scambiarsi informazioni attraverso un certo mezzo di trasmissione. Le reti di computer verranno trattate in grande dettaglio nel prossimo volume.
Molti S.O. del giorno d'oggi sono in grado di mettere le proprie risorse a disposizione di altri computer, ad essi collegati attraverso una rete.

Come un S.O. multiutente, un S.O. per server di rete deve gestire una lista di utenti e concedere l'uso delle risorse secondo i diritti di ciascuno degli utenti. A differenza di un sistema multiutente un server non esegue i programmi dei suoi utenti[2] ma concede ad essi solo l'uso delle risorse; se effettua elaborazioni esse non sono sotto il controllo diretto dell'utente "remoto" (client).

Alcuni di questi sistemi, classificabili senz'altro come "multiprogrammati", possono essere monoutente, perché non eseguono i programmi per gli utenti della rete, ma mettono solo a loro disposizione delle risorse (file, stampanti, altri dispositivi).
Per esempio, Windows NT - 2000 è un S.O. usato per i server di rete, che permette l'accesso protetto a tutte le sue risorse da parte di ogni utente in rete che ne ha il diritto. Esso peraltro esegue solo i programmi dell'unico utente che usa il computer su cui il S.O. gira.
Al contrario sui sistemi Unix, un utente "esterno" al computer che fa da server, collegato ad esso per mezzo della rete, è in grado di lanciare suoi programmi sul server.

2 nel panorama dei S.O. commerciali si possono trovare alcune eccezioni a questa affermazione!

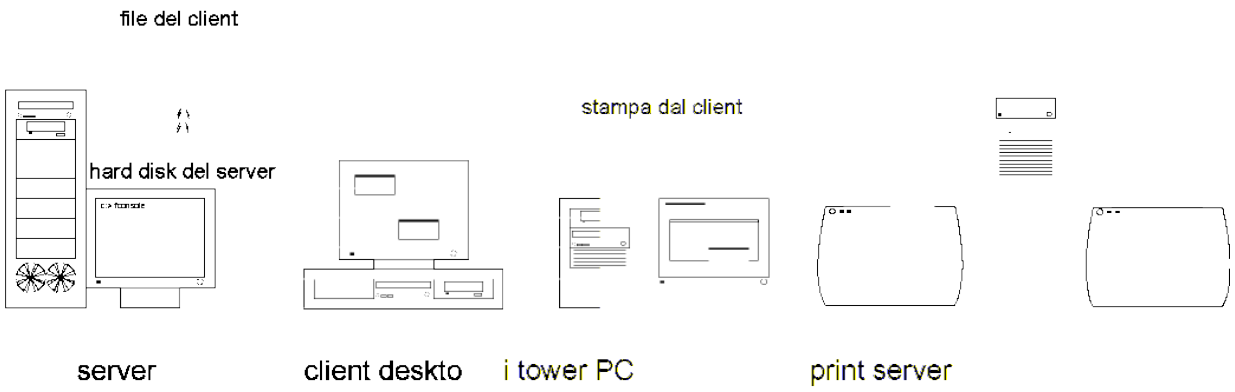


Figura 7: Server in rete locale

Application server
Vedi in seguito
!!!! todo !

Sistemi ad elaborazione distribuita (sistemi distribuiti)

Un sistema **distribuito** è costituito da diversi computer autonomi che, collegati attraverso una rete, sono in grado di concorrere, tutti assieme, ad eseguire parti della stessa elaborazione.

Un S.O. in grado di supportare l'elaborazione distribuita deve essere in grado di dividere "equamente" il lavoro dei computer in rete ("bilanciamento del carico", "load balancing") e di sincronizzarli quando si devono scambiare i risultati. In questo modo agevola la realizzazione dei programmi "distribuiti".
Ciascuno dei computer che partecipa ad un'elaborazione distribuita è autonomo; esso è perciò in grado di eseguire programmi per i suoi utenti senza il bisogno del supporto di altri sistemi. Peraltro quando uno di questi computer partecipa ad un'elaborazione distribuita esso eseguirà anche parte del programma "complessivo".

da fare !!!!
Figura 8: Sistema distribuito

S.O. multipiattaforma
Molti S.O. possono funzionare con CPU di famiglie diverse (p.es. Linux o le prime versioni di Windows NT). Questi S.O. vengono detti "**S.O. multipiattaforma**".
Quasi tutti i programmi scritti per un S.O. multipiattaforma sono in grado di girare su CPU di famiglie diverse semplicemente ricompilandoli o modificandoli solo in minima parte. Ciò conferma il fatto che l'astrazione dell'hardware fornita dai S.O. multipiattaforma è effettivamente indipendente dall'hardware reale che vi "sta sotto".

1.2 Caratteristiche dei Sistemi Operativi

Il S.O. mette a disposizione degli utenti di un computer un gran numero di **servizi**, dei tipi più disparati. Esso è un aggregato multiforme di moduli software che hanno funzioni e caratteristiche molto diverse. Le due funzioni principali che deve svolgere sono:
- gestione dell'interazione con l'utente ed esecuzione dei suoi programmi
- gestione delle risorse

Nei prossimi paragrafi si illustreranno alcune importanti caratteristiche dei Sistemi Operativi.

Interfaccia utente
Fanno parte del S.O. i programmi che permettono la comunicazione fra utente e S.O.. Viene detta "interfaccia utente" la modalità con la quale un S.O. si presenta ai suoi utilizzatori. L'interprete di comandi è la forma più semplice di interfaccia fra utente e S.O..
Un'interfaccia utente composta dall'interprete dei comandi e dai comandi stessi e che visualizza i risultati con "caratteri di tastiera", viene detta "interfaccia utente a carattere" (**CUI**: **C**haracter **U**ser **I**nterface, o **C**ommand **U**ser **I**nterface).
Un'interfaccia utente che fa uso di grafica a puntini (a "pixel", raster) viene detta **GUI** (**G**raphical **U**ser **I**nterface, interfaccia utente grafica). La gran parte dei S.O. odierni possiede una GUI a finestre, tramite la quale l'utente può svolgere i compiti fondamentali relativi all'uso del computer (lancio dei programmi, manipolazione dei file, gestione dei dispositivi).

Caricamento ed esecuzione dei programmi, interprete dei comandi

Il S.O. comprende un programma "loader" che legge i file dei programmi da lanciare, li carica in memoria e li fa eseguire.
Collegato al loader è l'interprete dei comandi (**shell**), che "capisce" i comandi che devono essere eseguiti e può concatenarli e lanciarli in un ordine stabilito dai file di "script" (esempio: i file .BAT di MSDOS o gli "shell scripts" di Unix).

Uno "**script** di Sistema" è un file eseguibile costituito di comandi per il Sistema Operativo; esso per eseguire ha bisogno di un interprete[3].

Con l'eccezione di pochi comandi, che sono compilati all'interno del "cuore" del S.O. anche la maggior parte dei comandi del S.O. sono normali programmi che vengono eseguiti in modo analogo a quelli dell'utente. Se per esempio si cerca nei directory di MSDOS si trova il file FORMAT.COM, o MORE.COM. Lo stesso accade per i file grep o more di Unix.
I comandi che sono inclusi nel Sistema Operativo risiedono in memoria sin dall'accensione del computer, per questo non esistono file su disco che li contengano (esempio: DIR o CD in MS-DOS, pwd o cd in Unix).

Protezione della memoria e delle istruzioni

In un sistema multiprogrammato girano molti programmi contemporaneamente.
Perché il computer sia affidabile si deve fare in modo che un errore in uno dei programmi non coinvolga né altri programmi né il Sistema Operativo.
Il Sistema Operativo, spesso coadiuvato dall'hardware, realizza una "protezione" della memoria, assegnando diritti di accesso a particolari "blocchi". Quando un qualsiasi programma tenta di modificare locazioni di memoria cui non ha diritto di accesso il S.O. interviene, interrompe e "termina" quel programma, prima che abbia il tempo di far danni ad altri.
La protezione si può estendere anche ad alcune istruzioni di programma, che possono essere riservate ai programmi "più importanti". Qualora un programma provi ad eseguire un'istruzione che non ne ha il diritto di usare esso verrà terminato dal Sistema Operativo.

Realizzazione di macchine virtuali

L'insieme hardware + Sistema Operativo mette a disposizione dell'utente funzioni più sofisticate di quelle possedute dal solo hardware. Dunque l'insieme hardware + software costituisce una "**macchina virtuale**" con funzionalità aggiunte rispetto al solo hardware. Seguono le descrizioni di alcuni esempi di "macchine virtuali".

Multiprogrammazione e multiutenza

Un singolo computer può eseguire molti programmi di molti utenti.
Ogni programma si comporta come se avesse una CPU tutta per sé, anche se la CPU è una sola. Dunque un'unica CPU, rivestita del software che realizza la multiprogrammazione, costituisce una macchina virtuale con molte "CPU virtuali".
Analogo discorso si può fare con la multiutenza. Ogni utente vede una macchina virtuale che gli appare come un computer a lui interamente dedicato. Con un S.O. multiutente un computer si trasforma in "molti" computer, uno per ogni utente.

Memoria virtuale

I programmi che eseguono in un S.O. con memoria virtuale possono ottenere dal S.O. l'allocazione di più memoria di quanta sia la RAM effettivamente presente sul computer. Una macchina virtuale che comprenda la memoria virtuale può utilizzare più memoria di quella che il computer ha fisicamente a disposizione.
Come questa funzione venga realizzata effettivamente verrà spiegato successivamente.

Virtualizzazione dell'I/O

I programmi dell'utente sono "isolati" dall'hardware e ne vedono una versione "astratta".
Essi possono ignorare tutti i dettagli dell'hardware, che sono noti solo al S.O. . E' il S.O. che si occupa di:
- programmare i circuiti elettronici di I/O del computer, scrivendo nei loro registri di stato
- effettuare tutto l'Input/Output dai dispositivi esterni, interrupt e DMA inclusi.
- gestire e proteggere le porte di I/O del computer

Nei S.O. Unix l'astrazione dell'hardware è tale da trasformare ogni dispositivo in un file. Per utilizzare un dispositivo i programmi d'utente Unix non fanno altro che scrivere o leggere da un file "virtuale", senza doversi interessare di alcun dettaglio hardware. L'astrazione dei dispositivi fisici porta dunque ad una semplificazione della loro utilizzazione.

Virtualizzazione dei computer

E' anche possibile, e molto in voga al momento, rendere virtuale tutta l'architettura di un intero computer, virtualizzando non solo la CPU, ma anche i circuiti di I/O, la scheda video, la tastiera ed in generale tutti i dispositivi.
La virtualizzazione dei computer rende possibile l'installazione di molti sistemi operativi sullo stesso computer fisico. Oguno di questi sistemi vede un computer virtuale che crede di avere a sua completa disposizione, mentre in verità l'hardware è usato a turno dai monlti computer virtuali che vi girano.

3 per la definizione di interprete si veda il volume precedente.

Queste tecnologie sono molto usate al giorno d'oggi quando si vuole dare la possibilità a molti utenti di configurare a piacimento la propria macchina, spendendo di meno in hardware. Esempi di tali situazioni sono i datacenter, le "server farm" per Internet, i laboratori di informatica delle scuole e delle università.
Da notare che la virtualizzazione dei computer permette di "isolarli" completamente fra di loro e se, per esempio per errore di configurazione, un S.O. non è più in grado di funzionare, ciò non influenzerà minimamente gli altri S.O. che girano sullo stesso hardware.

Virtualizzazione completa e paravirtualizzazione
I software che virtualizzano i computer lo possono fare a "basso livello", al di sotto di ogni S.O. installato sull'hardware, ed allora si parla di **virtualizzazione completa**. In questo caso i sistemi virtualizzati devono essere identici, anche nell'hardware virtuale all'hardware fisico esistente sul computer. Questo modo di operare dà origine a sistemi più efficienti, usati nelle applicazioni professionali, ma ha lo svantaggio che il software di virtualizzazione deve essere l'unico a controllare l'hardware, per cui può funzionare solo su quei device per i quali il produttore ha scritto i driver. In pratica il sofware di virtualizzazione completa è un "piccolo S.O." che controlla altri S.O. Esempi di tali software sono: VMware Server (programma commerciale) e XEN (programma libero).
L'altro approccio che viene usato è la "**paravirtualizzazione**", ove un'applicazione che gira in un S.O. ospite è in grado di simulare diversi computer. Questo modo di funzionare è senz'altro più lento, ma può senz'altro bastare per la simulazione di ambienti di rete e per il consolidamento di pochi server sullo stesso computer. Esempi di software che "paravirtualizzano" sono: la versione gratuita di VMware Server, Virtual Server di Microsoft (programmi commerciali) e User Mode Linux (programma libero).
Dal punto di vista operativo, il programma che gestisce la virtualizzazione dei computer viene eseguito nel S.O. "ospite", crea un grande file nel quale mantiene una "fotografia" di tutto quanto gli serve per emulare il nuovo computer e poi lo fa partire. Si apre una finestra che contiene quanto verrebbe visualizzato dal computer virtuale. Inserendo il DVD che contiene il Sistema Operativo desiderato lo si può installare sul computer virtuale e poi utilizzarlo come se fosse un computer completamente diverso da quello "ospite". Si può anche simulare una rete che colleghi i vari computer virtuali che sono presenti sul computer ospite.

1.3 Processi

A questo punto si deve introdurre il termine "processo" molto usato nella trattazione dei Sistemi Operativi multiprogrammati.

Un "**processo**" (process) è un programma nel momento in cui esegue in un S.O. multiprogrammato.

Come definizione di programma vale ancora quella data nel volume precedente: "un programma è un insieme di istruzioni che risolve un singolo problema".
Un processo è qualcosa di più di un programma, perché per eseguire un programma in un sistema multiprogrammato non è sufficiente possedere il suo codice.
Infatti in un sistema in multitasking ogni programma può essere interrotto dal Sistema Operativo, per farne eseguire un altro.
Dato che un programma può essere sospeso è indispensabile che il S.O. "ricordi" tutte le informazioni, che servono a riprenderne l'esecuzione in un secondo tempo.

Un processo (detto anche "**task**", che significa "compito", "missione") è perciò costituito dal codice del programma, dai suoi dati, e da un insieme di informazioni che servono a riprenderne l'esecuzione quando essa sia sospesa dal Sistema Operativo, per qualunque ragione.

Queste informazioni vengono memorizzate dal S.O. in una struttura dati in memoria, detta "**descrittore di processo**".
Fra questi dati i più importanti sono:
- l'indirizzo della prossima istruzione da eseguire, quando il processo verrà fatto ripartire
- il valore di tutti i registri della CPU al momento della sospensione del processo. Essi verranno ripristinati prima di riprendere l'esecuzione del processo.

Nel seguito si discuterà in maggiore dettaglio il contenuto del descrittore di processo.
E' importante rimarcare il fatto che "programma" e "processo" sono concetti diversi, anche se analoghi per molti aspetti. Per esempio è comune il fatto che uno stesso programma possa essere "usato" da processi diversi. La Figura 9 illustra appunto uno di questi casi.
I descrittori di processo di P1 e P2 contengono un campo che li fa puntare all'indirizzo al quale il processo riprenderà l'esecuzione, indicato come "ind. Codice" in Figura 9.
La situazione illustrata nella figura mostra il processo P1 in esecuzione. Si può dire che viene illustrato il momento esatto in cui P1 ha ripreso ad eseguire dopo una sospensione, infatti il Program Counter punta proprio all'indirizzo contenuto nel descrittore di processo.
I processi P1 e P2 evolvono indipendentemente uno dall'altro e percorrono il codice "comune" in tempi diversi.

Ogni processo ha il suo spazio di indirizzi autonomo, che gli altri processi non hanno diritto di usare, mentre può essere presente una memoria comune, sulla quale possono intervenire più processi.
Come illustrato dalla Figura 9 ciascun processo ha una parte di memoria allocata per il suo uso esclusivo, l'indirizzo della quale viene memorizzato nel descrittore di processo.
Nella stessa figura è indicata anche un'area di memoria comune, attraverso la quale i processi possono comunicare fra loro e con il Sistema Operativo. L'indirizzo dell'area di memoria comune non è indicato nel descrittore di processo perché si considera, come avviene in S.O. reali, che quell'area di memoria parta sempre dallo stesso indirizzo, fisso.

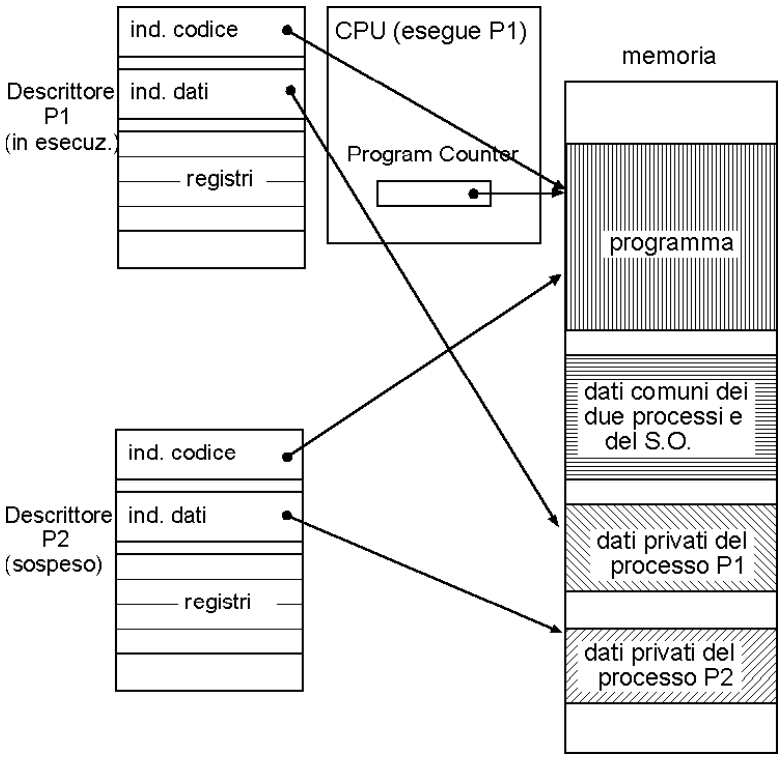


Figura 9: due processi condividono in memoria il codice dello stesso programma

1.4 Gestione delle risorse del sistema

Dato il concetto di processo, è giunto il momento di dare una definizione più rigorosa di "risorsa" in un S.O..

Si può definire "**risorsa**" una qualunque entità hardware o software che deve essere usata da un processo e ne condiziona l'avanzamento.

Una funzione fondamentale di un Sistema Operativo è gestire le risorse del sistema ed in particolare, nel caso di sistemi multiprogrammati, il rendere possibile la condivisione delle risorse fra i diversi processi in esecuzione.

Si possono individuare tre categorie di risorse da condividere:
- risorse hardware (es. CPU, memoria, periferiche)
- programmi (es. librerie condivise)
- dati (es. database, variabili comuni a più programmi)

Per gestire le risorse il S.O. deve:
- mantenerne aggiornato lo stato, gestendo un "descrittore" della risorsa, cioè una struttura dati che contiene al suo interno tutti i dati utili a rappresentare la condizione corrente della risorsa
- assegnare la risorsa ai processi che la richiedono, impedendo l'accesso ad altri processi se necessario
- realizzare una politica per la sua gestione, pianificandone l'assegnazione ("**scheduling**")
- toglierla al processo che l'ha richiesta quando esso non ne ha più bisogno, rendendola disponibile per altri processi

Il S.O. "crea" e/o gestisce le risorse in modo che:
- sia possibile accedervi
- la loro utilizzazione sia condivisibile da più di un utente
- il loro accesso sia assicurato solo a chi ne ha il diritto

Possono essere considerate risorse, per esempio: la CPU (o le CPU), la memoria, le periferiche, i file, i programmi.

Contabilizzazione dell'uso delle risorse (**accounting**)
Nei sistemi multiutente e nei server di rete il S.O. può tener traccia dell'uso che ogni utente fa di ogni risorsa.
Questo può servire per far pagare all'utente il servizio in base all'uso che ha fatto delle risorse (p.es. 10 m€/ms per ogni millisecondo di utilizzazione della CPU, 5 m€/MByte per l'occupazione della memoria).
Anche se non interessa far pagare all'utente il servizio, l'accounting può essere utile all'amministratore del sistema per capire come esso viene usato e per decidere meglio come modificarlo ed evolverlo.

Alcuni servizi del S.O.
In questo paragrafo elenchiamo alcuni dei servizi che ogni S.O. deve mettere a disposizione; altri servizi, meno visibili all'utente finale, saranno introdotti in seguito.

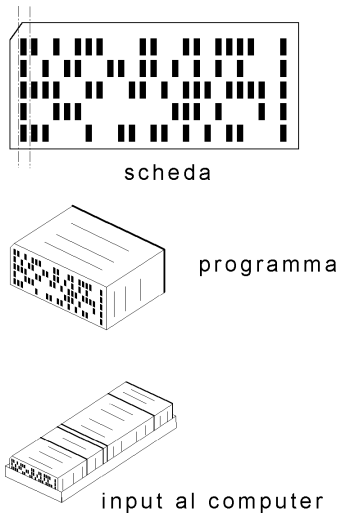
- garantire un accesso ordinato a periferiche, quali stampanti, dispositivi di memoria di massa, terminali, reti di computer
- realizzare un "file system"
Compito di un file system è realizzare il "concetto" di file organizzando e gestendo opportunamente la memoria di massa (hard disk); a questo scopo è necessario:
    - assicurare l'assegnazione e la manutenzione di un "nome" per ogni file
    - gestire un metodo per assegnare e tenere traccia delle caratteristiche dei file ("attributi" dei file)
    - organizzarli logicamente in "contenitori" ("directory" o "folder")
    - rendere possibile la loro lettura e scrittura, controllando nel contempo che l'accesso ai file sia possibile solo da parte di chi è autorizzato
- gestire i processi
Nella gestione dei processi è necessario che il Sistema Operativo metta a disposizione funzioni per:
    - creare i processi
    - assegnare la CPU ai processi (scheduling)
    - realizzare politiche di assegnazione delle risorse ai processi
    - distruggere i processi (terminazione come evento "normale" o per la presenza di un errore)
    - realizzare strumenti per la comunicazioni fra i processi (interprocess communications)
    - realizzare strumenti per la sincronizzazione dei processi
- comunicare con l'utente
    - identificare l'utente (logging "in" e "out")
    - realizzare un'interfaccia con l'utente: visualizzare i programmi, catturare l'input da tastiera e mouse
    - gestire dell'utente e addebitare i costi (accounting)
- rilevare e gestire gli errori, quali p.es.:
    - overflow ed altri errori nelle operazioni aritmetiche
    - risorse insufficienti
    - violazioni di "protezione" (istruzioni e/o accessi alla memoria non concessi)
    - violazioni di sicurezza (accesso non autorizzato a informazioni)

1.5 Un po' di storia

I S.O. dei primi computer
ENIAC (**E**lectronic **N**umerical **I**ntegrator and **C**omputer) è considerato il il primo computer operativo e pienamente funzionale, ed è senz'altro il primo che acquisì chiara fama. Sui computer come ENIAC le unità di ingresso ed uscita erano nastri cartacei che venivano perforati con i codici binari dei numeri. Come dispositivo d'uscita ENIAC aveva anche luci che si accendevano in base ai numeri binari che dovevano uscire. Queste luci sono state per decenni il "simbolo" dei computer nel sentire comune.
E' chiaro che i primi programmi scritti sui primi computer sono stati volti a migliorare un minimo il modo con cui l'utente interagiva con il computer (l'interfaccia utente).
Si trattava comunque di strumenti rudimentali; i primi computer non avevano un vero e proprio S.O., che "isolasse" l'utente dalle complicazioni dell'hardware, ma solo un programma, detto "**monitor**" che aveva solo funzioni di lancio dei programmi dell'utente e di minimo controllo delle applicazioni.
La sua operatività era simile a quella di un debugger non simbolico, si poteva lanciare il programma ed eventualmente tracciarne il comportamento "passo a passo".
Il computer era controllato esclusivamente da un'unica tastiera, detta "console", a disposizione di un unico operatore. Il programma veniva caricato, corretto ed eseguito direttamente alla console.

Elaborazione batch (a lotti)
Dati i costi astronomici dei primi computer fu subito necessario sviluppare del software "di sistema" in grado di far girare in modo automatico i programmi scritti da molti utenti diversi.
I monitor di generazione successiva erano in grado di lanciare automaticamente i programmi ad una certa ora della giornata; in questo modo fu possibile organizzare l'esecuzione "a lotti" (**batch**), durante tutto l'arco della giornata.
I programmi di utenti diversi venivano raggruppati e forniti al computer, che le eseguiva automaticamente in sequenza.

La modalità di utilizzazione dei primi sistemi batch era la seguente:
- Il programmatore scriveva il programma su schede di carta perforate. Per questo usava la "perforatrice" (card puncher), una macchina che praticava cinque fori in codice Hollerit per ogni carattere che veniva digitato sulla tastiera, in colonna sulla scheda, una colonna per carattere.
- Il programmatore ordinava le schede nella corretta sequenza, mettendo una scheda di colore diverso in fondo. Sulla scheda finale era perforato il codice di "end of file".
- Il programmatore consegnava il pacchetto di schede al Centro di Calcolo.
- L'operatore del Centro di Calcolo riponeva le schede in un cestello, di seguito alle schede di altri programmi.
- Il cesto veniva posto in un'unità di ingresso al computer: il lettore di schede perforate, che leggeva i codici Hollerit in base al fatto che la luce passava o meno dai fori. I numeri binari corrispondenti venivano passati al computer.
- Il computer leggeva tutte le schede del primo programma nel cestello, poi lo compilava e lo eseguiva.
- I risultati dell'elaborazione venivano stampati su carta da una stampante. Alla fine dell'esecuzione del primo programma il computer passava alla compilazione ed all'esecuzione del secondo.
- Alla fine del programma l'operatore raccoglieva le schede ed il tabulato con i risultati (o con la lista degli errori!) e li metteva in un cestello all'uscita del Centro di Calcolo.
- Alcune ore dopo aver lasciato le schede al Centro di Calcolo il programmatore andava a riprendersele, con il tabulato dei risultati, effettuava le modifiche su nuove schede, le rimetteva in ordine e ricominciava il processo.



Come si può ben capire il sistema era lento e macchinoso ed i programmi dovevano funzionare "alla cieca" senza alcun tipo di interazione con l'utente.
I sistemi batch di seconda generazione scaricavano programma e dati su nastro, per poter effettuare letture e scritture più rapidamente e per poter leggere i dati da più lettori di schede "contemporaneamente". Infatti mentre un lettore scaricava temporaneamente i programmi su un nastro, la CPU poteva eseguire i programmi scritti su un altro nastro, eventualmente da un altro lettore.

Figura 10: elaborazione batch

Nei sistemi batch i programmi venivano detti "**job**" (lavori).
Il termine è usato anche oggi, per indicare quei programmi che vengono eseguiti con regolarità e senza alcuna interazione con l'utente. Per esempio, nei sistemi odierni si chiamano "job" quei programmi per il backup o la manutenzione del sistema, che vengono lanciati automaticamente nel corso della notte.
L'elaborazione batch "pura" era un sistema poco efficiente nell'uso del computer, dato che la CPU doveva perdere molto tempo per attendere che la stampante, molto più lenta, producesse i risultati.

Spooling
Un miglioramento dell'efficienza avvenne con l'introduzione di processori di I/O dedicati, che permettevano di "dilazionare" il momento della stampa e di suddividerlo su più di una stampante.
Questi sistemi, che servivano solo per le stampanti, erano in grado di immagazzinare i dati spediti dalla CPU per la stampa e trasferirli a tempo debito ad una delle stampanti collegate.
In questo modo si poteva avere una sovrapposizione temporale dell'attività di I/O con quella della CPU, aumentando l'efficienza del sistema.
La tecnica in questione venne chiamata **spooling** (**S**imultaneous **P**eripheral **O**perations **On** **L**ine, più periferiche collegate "on line" contemporaneamente). Il termine spooling è ancora usato per indicare tutte le tecniche che prevedono la stampa "dilazionata" (detta anche stampa in "background").
Al giorno d'oggi ciò avviene tipicamente attraverso un software, in genere facente parte del S.O., che cattura tutti i dati che un programma utente invia ad una stampante, li memorizza in una "**coda dello spooler**" sull'hard disk, poi li trasferisce alla stampante "con comodo", rispettando i suoi tempi, molto più lenti rispetto a quelli della CPU.
Il programma utente viene così liberato molto rapidamente dalla necessità di stampare. Per il programma utente non c'è differenza nella stampa sullo spooler o direttamente sulla stampante, lo spooler viene semplicemente visto come una stampante velocissima.

Sistemi centralizzati interattivi

Dato il costo dei primi computer le CPU erano poche e dovevano essere concentrate in un unico punto nel quale erano facilmente accessibili da personale qualificato.
Quei computer erano giocoforza dei "sistemi centralizzati", con un "Centro di Calcolo" nel quale si trovano tutte le CPU e le principali periferiche.
Nacque presto l'esigenza di fare in modo che l'utente potesse intervenire sul programma da una postazione distante dal Centro di Calcolo e mentre il programma era in esecuzione.
Entrambe queste esigenze furono risolte con i **terminali interattivi**[4].

4 La parola "interattivo" significa che l'utente può colloquiare con il programma (inter-agire) mentre esso sta eseguendo.

I terminali sono dispositivi separati dall'unità centrale, eventualmente anche molto lontani, e permettono agli utenti di lanciare i programmi, di vederne immediatamente i risultati e di intervenire sul loro funzionamento mentre stanno eseguendo.
I terminali possono essere collegati all'unità centrale in diversi modi, i più tipici sono: collegamento seriale, collegamento in rete locale, collegamento via modem.
Un terminale può essere dislocato in prossimità della unità centrale o può essere anche a grande distanza (terminale remoto). Con un collegamento telefonico via modem è possibile che un terminale sia a qualsiasi distanza dall'unità centrale, in un qualsiasi punto della terra.

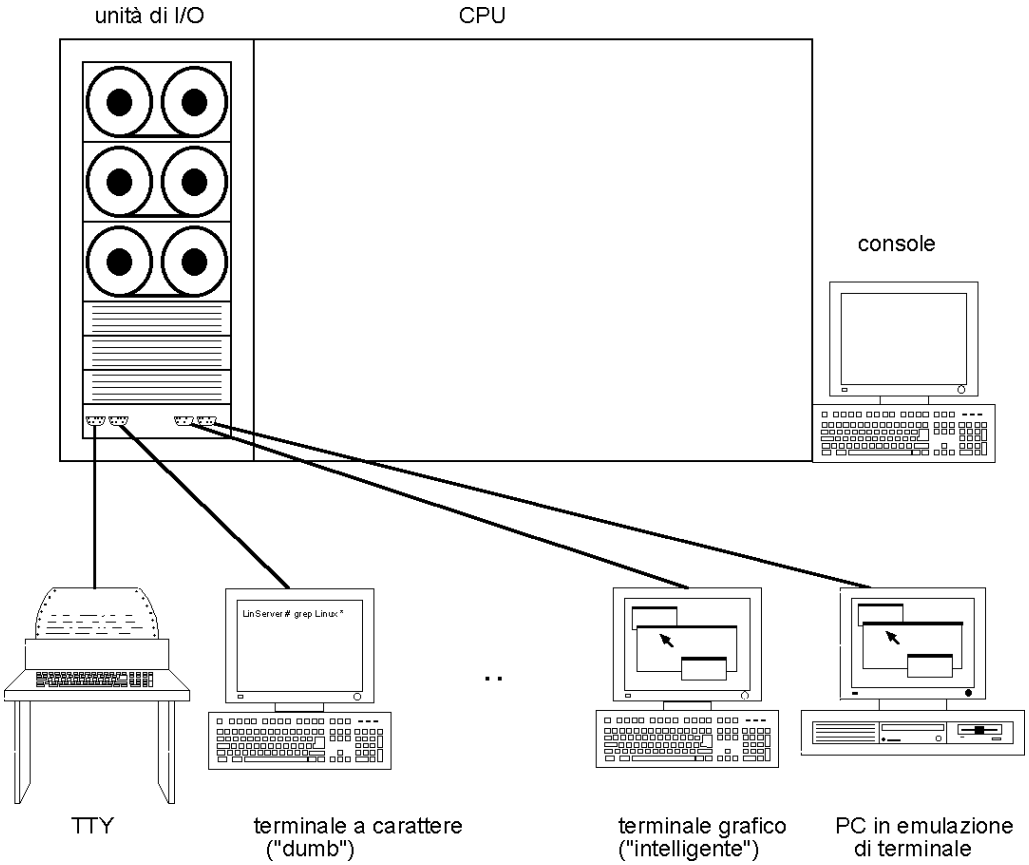


Figura 11: mainframe

I sistemi centralizzati sono ancora in uso, anche se oggi convivono con architetture distribuite e sistemi in rete locale.
Ciò significa che anche in tempi in cui le CPU vengono inserite negli orsacchiotti giocattolo può ancora essere opportuno centralizzare le risorse elaborative.
Infatti la centralizzazione può significare, se ben gestita, risparmi nella manutenzione dei sistemi e nell'ottimizzazione della loro utilizzazione.
Il vantaggio principale dei sistemi centralizzati risiede comunque nella loro affidabilità e nella sicurezza. Si tratta infatti di sistemi il cui software di base non è cambiato radicalmente nel corso di quasi trent'anni, per cui è molto ben collaudato ed assolutamente affidabile.
Inoltre l'hardware per i sistemi centralizzati è pensato per un servizio affidabile e continuativo, 24 ore su 24, 365 giorni all'anno. Lo svantaggio più evidente è che ha un rapporto prezzo/prestazioni molto più svantaggioso rispetto ai personal computer.

I mainframe

Il centro del Centro di Calcolo è il "**mainframe**"[5], cioè il computer centralizzato, nel quale si concentra la capacità elaborativa del sistema.
Per erogare questa capacità elaborativa a molti utenti era necessario che il Sistema Operativo fosse multiprogrammato e multiutente.
La modalità di elaborazione "batch", precedentemente descritta, non si adattava al lavoro con il terminali interattivi, perché poteva lasciare l'utente senza risposta da parte del sistema per lunghi periodi.
Fu dunque necessario fare in modo che l'utente al terminale potesse lavorare come se avesse a disposizione la CPU solo per sé. Questa funzionalità fu realizzata nei sistemi "time sharing".

5 In Inglese la parola "frame" significa anche "intelaiatura" o "scatola", l'apposizione del prefisso "main" ("principale") è comprensibile ..

Time-sharing (divisione di tempo)
In un sistema time-sharing il tempo della CPU viene condiviso (time share = condivisione di tempo) fra i vari processi interattivi.
Essi hanno diritto ad eseguire solo per un certo tempo, detto **time slice** ("fetta" di tempo :-) o **time quantum.** Scaduta la time slice il S.O. prende il controllo e sospende il processo che sta girando, assegnando la CPU ad un altro processo.

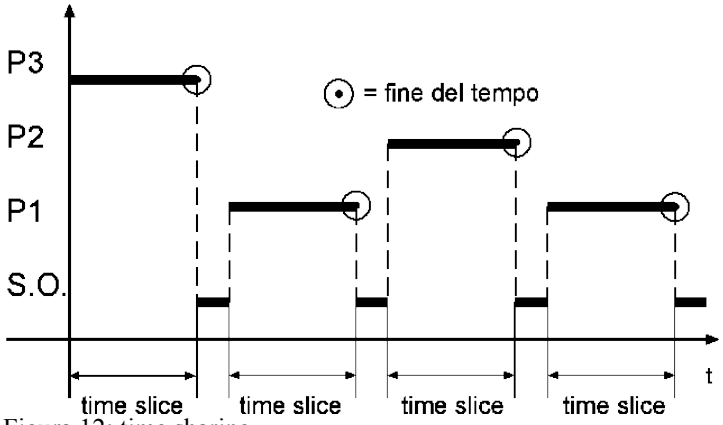


Figura 12: time sharing

Time sharing e realtime clock
Un dubbio che potrebbe cogliere il lettore a questo punto è il seguente. Come fa un processo d'utente a capire che è finito il tempo che ha a disposizione e che il S.O. vuole togliergli la CPU, per darla ad un altro processo?
Se la CPU è una sola il S.O. non può eseguire mentre esegue il processo d'utente e perciò non può comunicare alcunchè al processo.
E' forse il programma d'utente che deve andare a verificare regolarmente un orologio hardware per sapere se è ora di lasciare la CPU? Questo modo di operare avrebbe limitazioni molto gravi, sia per le complicazioni che aggiungerebbe ai normali programmi d'utente, sia per la bassa efficienza del sistema che ne deriverebbe.
La risposta è semplice se ci si fa aiutare dall'hardware. Per realizzare un sistema time-sharing è indispensabile che l'hardware comprenda un interrupt da un timer.
L'interrupt del timer viene lanciato regolarmente, allo scadere di un intervallo di tempo prefissato, che potrebbe anche non coincidere con una time slice (sappiamo per esempio che nel PC l'interrupt del timer viene lanciato 18,2 volate al secondo).
La procedura di risposta all'interruzione del timer fa parte di un programma molto importante del S.O., detto "process scheduler" (vedi oltre).
Dunque ad ogni interrupt del timer il processo corrente viene interrotto e viene eseguito il "process scheduler", che verifica se il processo appena interrotto ha terminato o meno la sua time slice.
Se il tempo non è terminato l'interrupt si conclude con una semplice IRET, determinando la ripresa dello stesso processo interrotto, se invece il "time slice" è concluso il S.O. (processo scheduler):
- sospende il processo corrente, salvandone lo stato nel suo descrittore
- decide quale altro processo dovrà sostituire quello interrotto, secondo una sua ben precisa "politica"
- mette in esecuzione il processo stabilito.

In questo modo il programma utente non ha nessun bisogno di sapere che è finito il tempo a sua disposizione, perché a togliergli il possesso della CPU ci pensa il Sistema Operativo, coadiuvato dall'interrupt del timer.
Senza l'intervento dell'interrupt del clock sarebbe impossibile realizzare un sistema in time-sharing, perché in quel caso l'unico modo per togliere la CPU al processo che la possiede sarebbe una richiesta di "autosospensione" fatta dal processo stesso.
Vedendo la cosa dal punto di vista del processo, esso non si accorge nemmeno di essere stato sospeso, perché viene "congelato" mentre sta eseguendo un'istruzione qualsiasi e riprende ad eseguire dopo, un po' di tempo, nelle medesime condizioni in cui era stato congelato.

Una buona scelta della durata della time slice può avere grande influenza sulle prestazioni del sistema e sulla sua facilità d'uso.
La scelta di un tempo piccolo significa che i terminali interattivi sono serviti più spesso. Ciò darà all'utente un'impressione più gradevole, perché il terminale "risponderà" più rapidamente a ciò che viene digitato da tastiera.
Naturalmente il passaggio da un processo ad un altro non è "gratis", per cui la scelta di un tempo di slice troppo piccolo potrebbe implicare un overhead eccessivo, che fa calare l'efficienza del sistema.

I minicomputer
!!!! to be done !

Supercomputer

I computer più veloci nell'elaborazione dei dati vengono detti **supercomputer**.
In passato sono esistiti supercomputer con una sola CPU. In tali computer sia la tecnologia elettronica che ogni dettaglio architettonico e costruttivo erano specificamente pensati per massimizzare le prestazioni di elaborazione ("number crunching").
Peraltro ci si è resi conto, già da diverse decine di anni, che per avere velocità di elaborazione davvero alte non bastava solo fare CPU ottimizzate per la velocità, ma bisognava eseguire i programmi contemporaneamente su più CPU. E' nato dunque il concetto di "**elaborazione parallela**", e si sono introdotti come supercomputer i primi sistemi multiprocessore.
Al mondo esistono supercomputer costituiti da decine di migliaia di CPU, che al loro interno possono far eseguire contemporaneamente anche quattro Sistemi Operativi diversi, che si suddividono e si contendono le CPU a disposizione.

Workstations

La realizzazione di circuiti integrati sempre più potenti rese possibile l'integrazione su singolo chip di intere CPU della potenza di quelle dei minicomputer. Ciò portò alla nascita delle workstation, computer potenti e multiutente, che di solito vengono usati da pochi utenti per compiti che richiedono grande capacità di elaborazione, oppure da molti utenti, ma come server in una rete.
Per le workstation vennero prodotti i microprocessori RISC, cui si accenna in altra parte di questo testo.

Unix

Unix è un S.O. sviluppato da Dennis Ritchie e Ken Thompson agli AT&T "Bell labs"
Fu scritto con l'intento di mantenerlo facilmente "portabile" su molte piattaforme hardware.
Ciò successe effettivamente ed Unix è il primo S.O. che si può dire che abbia girato su tutte le piattaforme, esclusi (forse!) i personal computer a 8 bit ed alcune console per videogiochi.
Il primo computer su cui Unix fu sviluppato fu un "glorioso" minicomputer, il DEC PDP 11. Presto i S.O. di tipo Unix si frammentarono in molti "dialetti" (alcuni dicono "Unix flavours", "gusti" di Unix :-), differenziati per la piattaforma e per il produttore, che ne ostacolarono l'espansione sul mercato.
Peraltro le similarità fra le varie versioni di Unix sono al giorno d'oggi molte, rafforzate anche dalla ratifica di standard quali "Posix", per cui lo sviluppo di applicazioni per Unix diversi non è molto complicato.
Tutto il mercato delle workstation scientifiche - grafiche e dei grossi server di rete è basato su S.O. "Unix like".

I personal computer

Anche i S.O. per personal computer hanno percorso un'evoluzione analoga a quella del S.O. destinati ai computer più grossi.
!!!! to be done !

I PC

!!!! to be done !

MS-DOS

!!!! to be done !

Windows

!!!! to be done !

Linux

!!!! to be completed !
Molti appassionati, disponendo di hardware diverso dei sorgenti e dell'ottimo gcc (GNU C Compiler) stabilirono diversi gruppi che realizzarono il porting del sistema sull'hardware più disparato. Tra le principali piattaforme ricordiamo:
Digital Alpha
ARM e Strong ARM (come i nuovi Corel Netwinder o svariati modelli della Acorn Computer)
Motorola 68K (per i vecchi MAC, Amiga e Atari ST)
PowerPC (Power MAC)
SPARC e Ultrasparc (anche con la benedizione di Sun)

Le reti locali

!!!! to be completed !

Applicazioni "peer to peer"
La gran parte della potenza dei microprocessori del giorno d'oggi resta inutilizzata; le uniche applicazioni di uso generale in grado di mettere alla frusta un computer odierno sono i videogiochi.

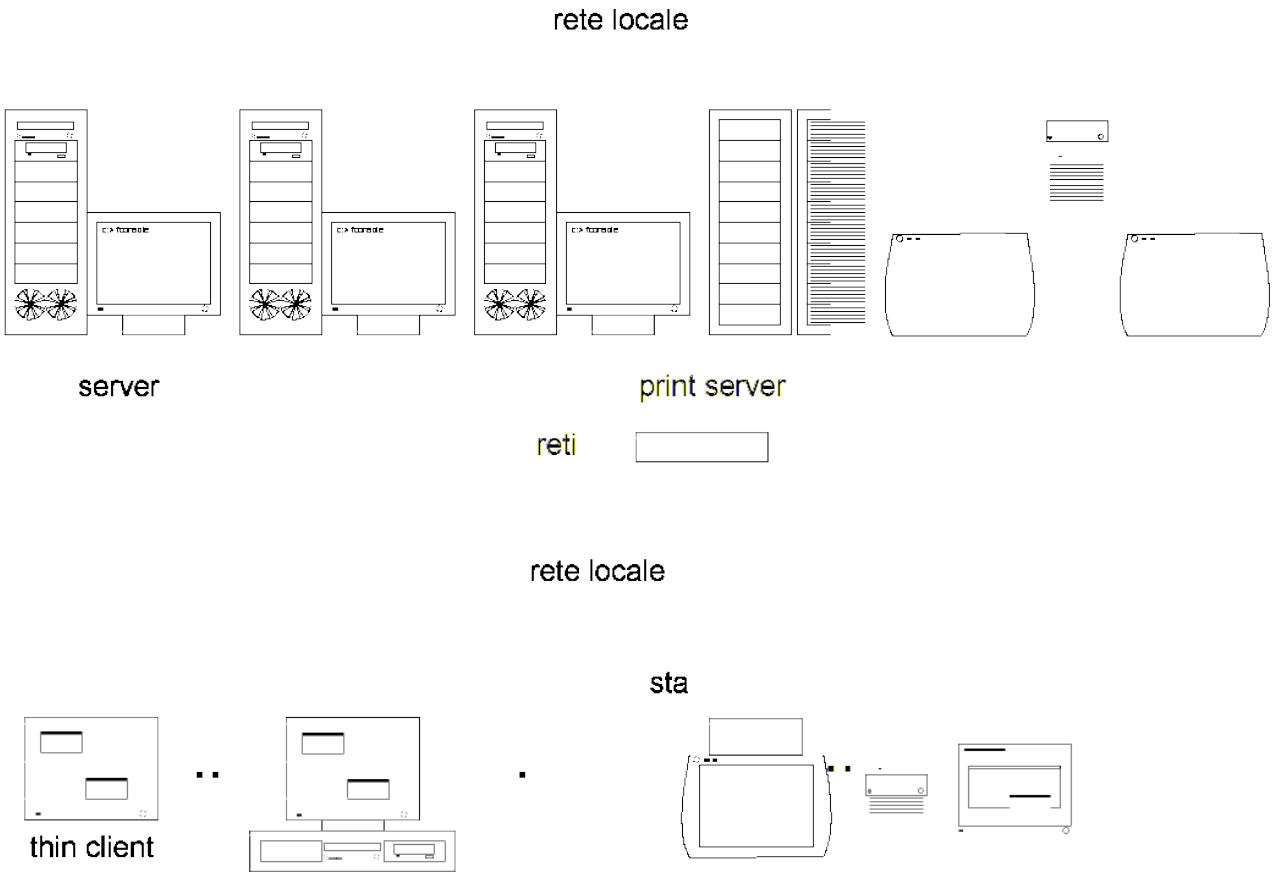


Figura 13: Sistema a server farm

Internet ed i computer ovunque
!!!! to be completed !!!!

1.5.1 Terminali

Il terminale è il più semplice dispositivo di I/O dei sistemi centrali multiutente. Prima dell'avvento dei personal computer (metà degli anni '70) erano l'unico modo con il quale si poteva usare un computer "interattivamente".

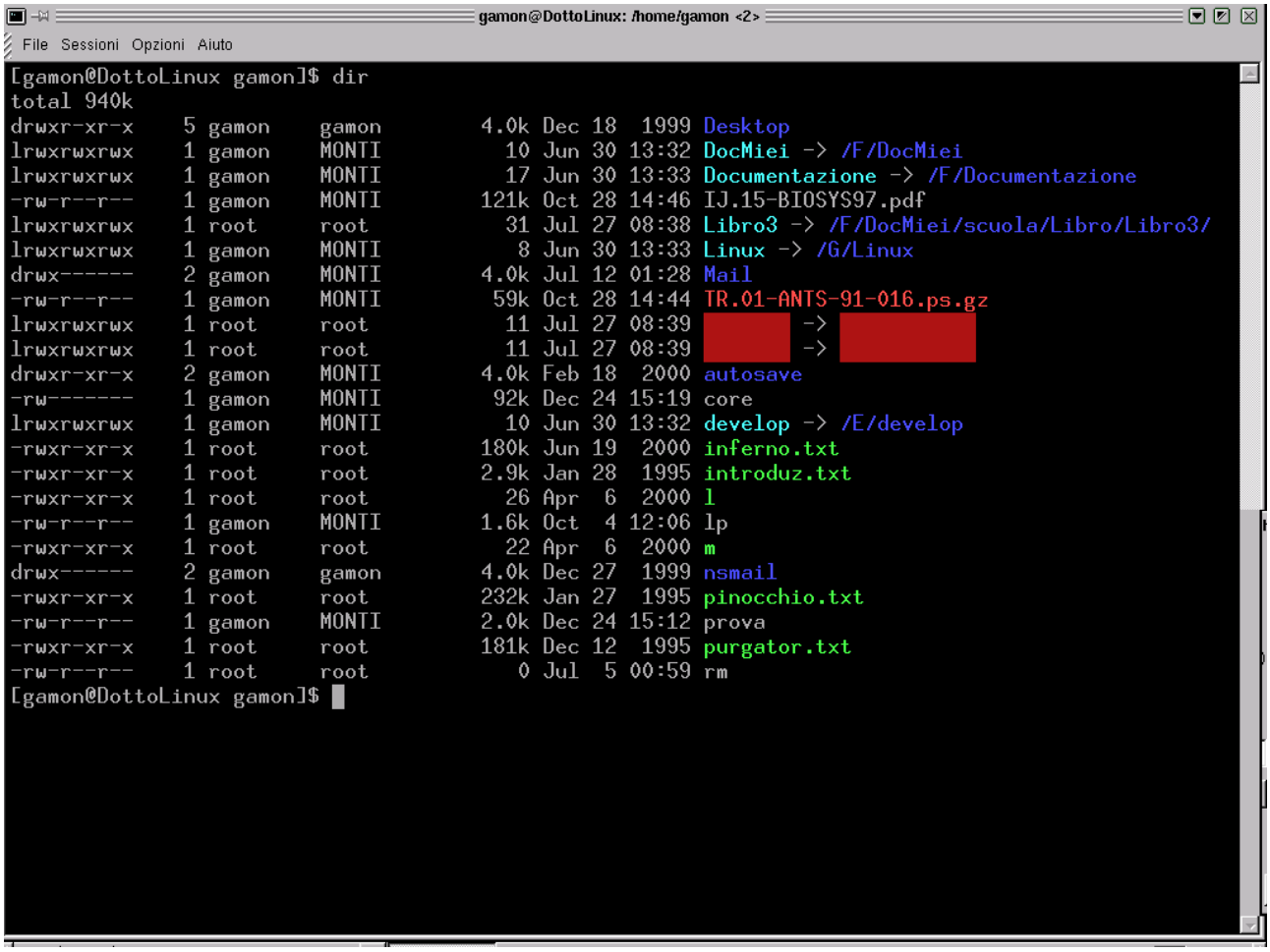
Telescriventi
Una telescrivente (**TTY**: **t**ele**ty**pewriter) è una tastiera con stampante, che non ha CPU ed usa le tecnologie elettromeccaniche della macchine per scrivere.

La telescrivente è stata per lungo tempo il mezzo attraverso il quale venivano spediti i telegrammi. Una telescrivente cattura i caratteri digitati sulla sua tastiera e ne trasferisce il codice ASCII ad un'altra TTY, collegata ad essa con una rete analoga alla rete telefonica. La telescrivente che riceve il carattere lo scrive sulla sua stampante. E' così possibile la trasmissione bidirezionale di testo da un capo all'altro del collegamento, fra punti qualsiasi della terra.

Il primo tipo di terminale interattivo utilizzato sui computer fu la telescrivente, che già esisteva e che non fu difficile adattare all'uso con un computer. Invece di spedire i codici ASCII ad un'altra TTY, il terminale TTY li manda all'unità di I/O del computer. La CPU acquisisce in questo modo i dati battuti sul terminale.
I risultati delle elaborazioni della CPU venivano poi spediti alla stampante della TTY, che così li poteva mostrare interattivamente.

Terminali a caratteri ("text terminals")
L'uso della stampante per ogni dato in uscita dal computer fu presto sostituito da un monitor a CRT, analogo a quello della TV.
Un terminale di testo è costituito da una tastiera e da un video: Esso non ha senza capacità elaborative, ma solo di Input/Output, è collegato all'unità centrale con un'interfaccia seriale (RS 232, vedi il prossimo volume), con un modem (anch'esso trattato nel prossimo volume), o con un sistema seriale sincrono quale SNA (terminali per mainframe IBM).
Ciò che viene digitato nella tastiera viene spedito all'unità centrale che lo riceve ed elabora, poi lo rispedisce al terminale, che lo visualizza sul video, come fa con tutto ciò che l'unità centrale spedisce.
Per la visualizzazione è necessaria un piccola quantità di memoria, che deve "ricordare" cosa visualizzare fra un passaggio e l'altro del pennello elettronico.

I primi terminali video a caratteri erano dispositivi di I/O che, come le TTY, non avevano nessuna capacità elaborativa autonoma. Per questo vengono chiamati "**dumb terminals**", "**terminali stupidi**".
Con i primi terminali stupidi si poteva interagire solamente con i caratteri alfanumerici della tastiera; in seguito vennero codificate delle particolari sequenze di caratteri che per il terminale erano comandi del tipo: "visualizza i prossimi caratteri in campo invertito, oppure lampeggiante o con certi colori".
In questo modo la qualità dell'output dei computer aumentò notevolmente ma ai terminali fu richiesta una, sia pur minima, capacità di elaborazione, per capire i comandi che venivano loro inviati.



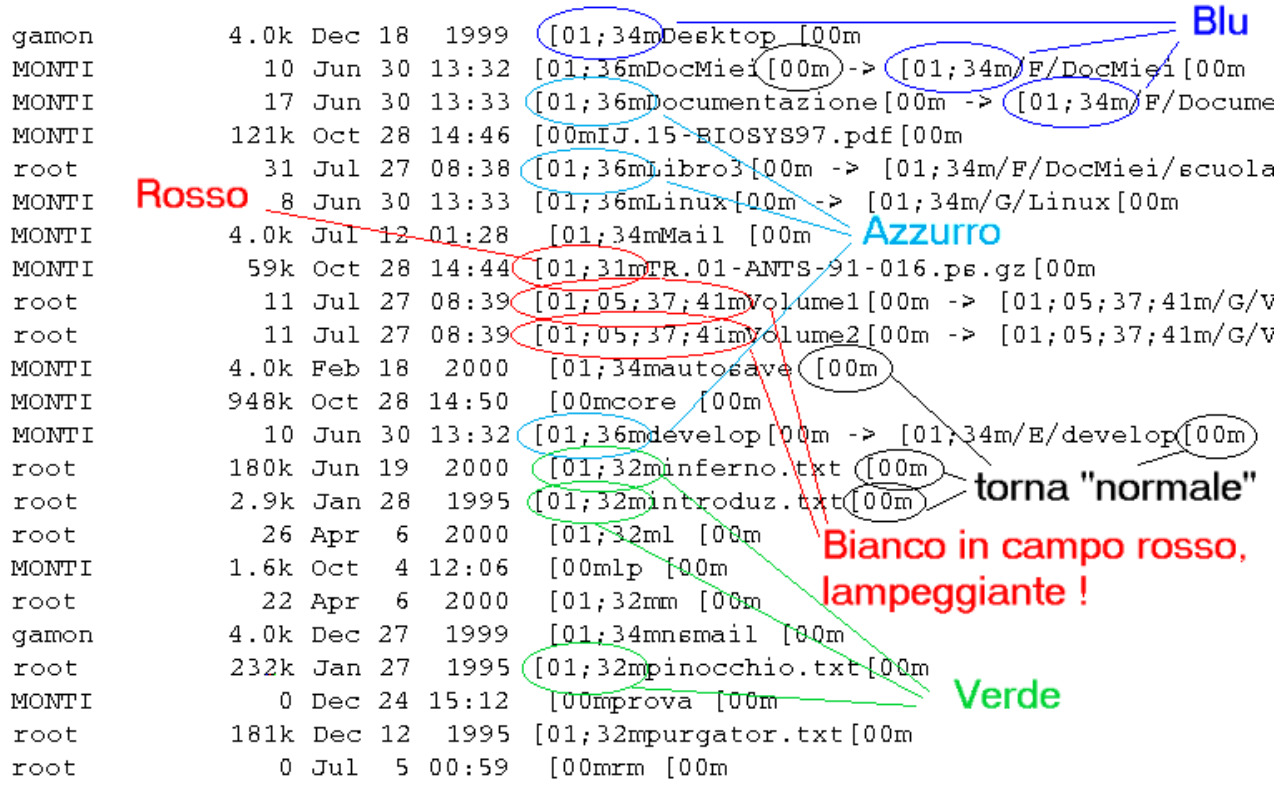


Figura 14: (a) ciò che viene visualizzato in un terminale a colori; (b) ciò che viene trasmesso dalla CPU

I comandi per i terminali sono costituiti da particolari sequenze di caratteri, dette "**sequenze di escape**", i terminali più moderni hanno centinaia di queste sequenze.

La Figura 14 (b) mostra i caratteri effettivamente spediti quando viene visualizzata la schermata in Figura 14 (a).
Si può vedere come le sequenze di escape che codificano i colori siano spedite per far "cambiare di stato" la visualizzazione, che prosegue con il nuovo colore fino a che non viene spedita la sequenza di escape che fa tornare al colore "normale".
Ogni produttore di terminali inventò la sua serie di sequenze di escape, per cui il S.O. deve sapere che tipo di terminale è collegato ad ogni sua porta, per poter spedire le sequenze di escape giuste.
Il tipo di terminale più usato fu il DEC VT100, che veniva anche emulato da terminali di altre marche. Questo terminale aveva alcuni tasti speciali e di funzione che venivano usati nei programmi interattivi.
Molti tipi di terminali IBM usavano il codice alfanumerico EBCDIC, che veniva utilizzato nei mainframe, in luogo del codice ASCII, usato invece in tutti gli altri terminali.

Terminali grafici

I terminali grafici sono in grado di visualizzare rapidamente grafica a pixel[6]. Normalmente possiedono anche un mouse per l'input in modo grafico. I monitor grafici ottengono dall'unità centrale "mappe di pixel" ("bitmap"), che scrivono nella loro memoria video per poter effettuare la visualizzazione.
Dato che si deve spedire il codice di ciascun pixel da visualizzare il traffico di dati verso un terminale grafico è molto più alto che nel caso di terminali di testo. Per questo molti terminali grafici sono collegati con interfacce di trasmissione più veloci della normale RS232, che usano doppini schermati ("twinax") o cavi coassiali.
Il tipo più sofisticato di terminale grafico è Xwindow (senza s finale!). Esso è uno standard per terminali nato in ambiente Unix, ma che può essere usato anche in altri S.O. .
XWindow prevede la spedizione di comandi grafici a più alto livello, meno prolissi della descrizione a pixel dell'immagine, facendo "risparmiare" un poco sulla velocità di connessione richiesta.

Invece di dire "colora il pixel (A,B) di verde, il pixel (C,D) di rosso" e così via per ogni pixel dell'immagine, un comando XWindow potrebbe suonare così: "disegna un rettangolo riempito di rosso delle dimensioni X e Y, a partire dal punto (W, Z).

6 un pixel è un punto colorato sullo schermo del monitor. Per ottenere una grafica a tutto schermo è necessario avere una memoria video nella quale è mantenuto un numero che stabilisce il colore di ogni pixel da memorizzare. La CPU o la scheda video devono calcolare e trasmettere alla memoria video un codice per ogni pixel dello schermo.

L'esecuzione di questi comandi richiede la presenza di una CPU nel terminale; essa peraltro non è coinvolta nell'esecuzione del programma, che avviene solo nel computer centrale, ma solo nella visualizzazione dei risultati.
I terminali per Xwindow sono dispositivi piuttosto costosi, dato che debbono includere tutto ciò che è normalmente già presente in un PC e richiedono un collegamento con l'unità centrale ad alta velocità. Per questo non sono molto usati e si preferisce usare programmi di emulazione del terminale Xwindow che funzionano su un normale PC.

Computer in emulazione di terminale

L'avvento del personal computer spazzò via il mercato dei terminali. I primi personal computer avevano capacità elaborative autonome e costo confrontabile, se non inferiore, a quello dei terminali dell'epoca. Per questo le aziende preferirono acquistare personal computer e dotarli, per il collegamento con in mainframe, di software di "emulazione di terminale". Un programma di emulazione di terminale gira su un PC, ma si comporta esattamente come se fosse un terminale, più o meno stupido. Equipaggiato con un programma di emulazione di terminale un personal computer si comporta nei confronti del mainframe esattamente come se fosse un terminale.
Su PC esistono programmi di emulazione sia per i terminali a caratteri che per quelli grafici o XWindow; un programma di emulazione molto semplice viene fornito con i S.O. Windows (Winterminal).
Lo svantaggio principale è il fatto che usando programmi di emulazione la tastiera è diversa da quella del terminale originale. I tasti di funzione dei vecchi terminali vengono fatti corrispondere ad alcuni caratteri speciali del tastierino numerico del PC. Per i terminali complessi è utile usare una maschera sulla tastiera che indica quali caratteri speciali e tasti di funzione sono fatti corrispondere a tasti del PC.

Windows Terminal Services

Nelle versioni per server più recenti del S.O. Windows è possibile definire "terminali virtuali" che eseguono programmi sul server, per conto di PC remoti, accessibili attraverso una rete locale. Il PC remoto ha solamente funzioni di visualizzazione,
mentre tutta l'elaborazione viene eseguita sul "server", che diventa in questo caso il computer centrale di un sistema multiutente a tutti gli effetti.
Il "client" di Windows Terminal Services è quindi di fatto un PC in emulazione di terminale, anche se, in modo un po' improprio, viene chiamato appunto "client".
Il vantaggio di avere un sistema con terminali Windows sta nella gestione centralizzata dei programmi e nella possibilità di usare come terminale computer vecchi, che non sarebbero in grado di far girare "da soli" le applicazioni più moderne e "pesanti".
Per l'uso dei servizi di terminale Windows è necessario pagare una licenza per ogni terminale attivato sul "server". Dal lato "client" si paga invece la sola licenza del S.O. (se si usa Windows; dato che esistono in Linux software gratuiti di emulazione di terminali Windows, se si usa Linux non si paga la licenza del S.O. "client").

Terminali telnet

Tutti i sistemi multiutente odierni sono in grado di far funzionare i propri programmi spedendo i loro dati su una connessione "virtuale" realizzata utilizzando i protocolli di Internet (TCP/IP).
In questo modo la funzione di terminale può essere svolta da qualsiasi computer collegato attraverso Internet e che utilizzi il protocollo specifico chiamato "**telnet**".
Se un computer multiutente è abilitato ad usare il protocollo telnet un qualsiasi utente collegato ad esso attraverso Internet può eseguire i propri programmi sul computer ospite, a patto naturalmente che disponga di un nome utente e di una password validi sul quel computer.

DA FARE !!!!

Figura 15: Software telnet (puTTY) collegato da Windows ad un computer su cui esegue Linux

"Neoterminali" (thin client, network computer, ultra thin client)

Recentemente il modello di elaborazione centralizzata ha subito un rinnovato interesse, dovuto alla sempre maggiore difficoltà di gestione dei PC.
Quando una grande Azienda deve gestire migliaia di personal computer l'assistenza e la manutenzione sono un grosso problema. Solamente il cambio di versione di un programma usato da tutti implica l'installazione del nuovo software su tutte le macchine.
Il modello dei "neoterminali"[7] prevede un terminale intelligente, o un computer "leggero", che si collega, via rete locale od Internet, ad un server ("application server") che esegue le applicazioni.
Il terminale, o "**thin client**" (client leggero (sottile)), non ha unità di memorizzazione locali (né floppy disk, né hard disk) e provvede solo alle visualizzazioni ed ad altri lavori di "housekeeping", mentre le applicazioni girano sul server.
L'application server spedisce al client le "videate" dell'applicazione e l'utente al terminale lavora come se essa risiedesse sul suo computer.
I vantaggi di questo approccio stanno nel fatto che basta aggiornare l'applicazione sul server per fornire a tutti gli utenti l'aggiornamento; inoltre gli utenti non possono manipolare il proprio computer, aggiungendo applicazioni o facendo pasticci che spesso richiedono l'intervento dell'assistenza.

7 Il termine è dell'autore, **non** è usato nella letteratura tecnica!

Ultimo fatto interessante è che con un thin client l'utente può far eseguire solo le applicazioni che è autorizzato ad eseguire e non può "perder tempo" con programmi "estranei".
Analogo è il discorso per i cosiddetti "network computer". Anche il network computer ha poche o nessuna unità di massa locale, i suoi programmi sono scaricati dalla rete in un linguaggio di alto livello (es. Java), in modo da velocizzare il tempo di scaricamento, ed eseguiti localmente dalla macchina virtuale Java.
I network computer non sono terminali, dato che il programma viene eseguito localmente, ma hanno le stesse caratteristiche dei terminali per quel che riguarda la gestione, perché i programmi sono memorizzati in remoto.

Web applications
Un modello di elaborazione molto applicato attualmente è quello delle "Web applications". In questo caso i programmi vengono eseguiti su un server che rispetta il protocollo del Word Wide Web (HTTP).
I risultati del programma ed i dati dell'utente vengono forniti attraverso un programma "client HTTP", cioè attraverso il browser con il quale si fa normalmente la navigazione Web (es. MS Internet Explorer, Mozilla, Opera ed altri).
Il programma esegue dunque dal lato del server HTTP, che diventa una sorta di application server, e fornisce al client (browser) pagine scritte nel linguaggio HTML, che il browser interpreta e visualizza.
Il programma "lato server" elabora i dati forniti dall'utente e produce pagine Web diverse in base ai dati ricevuti (pagine "dinamiche").

Controllo remoto, VNC
Server, client per il controllo remoto
compressione, Internet
Tutto il carico computazionale è concentrato sul server, il client deve solo essere in grado di trasferire dati dalla rete, di passare i pixel al display e di acquisire i dati dalla tastiera.
VNC

Troubleshooting remoto

Ultra thin client
L'utilizzazione di display e tastiere remote collegate in rete viene definito da qualcuno "Ultra Thin Client" (UTC), per via delle risorse estremamente limitate che richiede dal lato del client. Infatti, se non è attivata la compressione, il client deve solo leggere i dati dalla rete rispettando il protocollo VNC e trasferire i pixel al display. Dispositivi del genere sono usati, per esempio, nei display delle casse dei supermercati.
In ambiente industriale sistemi del genere sono apprezzati perchè si può mettere il computer in un ufficio ed il display nella fabbrica. Quindi solo il display dovrà essere particolarmente robusto, insensibile all'umidità alla polvere od alle temperature estreme, mentre il computer potrà essere del tutto "normale", e costare molto meno.
Un UTC si presenta come una "scatola" con una porta RJ45 per il collegamento in rete Ethernet, un connettore per il monitor VGA ed uno per la tastiera, che contiene una CPU davvero minimale, il cui compito è solo quello di realizzare il protocollo di controllo remoto (es. VNC). Esistono microcontrollori che includono al loro interno tutto l'occorrente per realizzare un sistema del genere, contenendo i costi in maniera molto significativa.

La console
La console è un terminale molto particolare, in quanto è collegato direttamente all'unità centrale. I messaggi più importanti che il sistema comunica all'operatore vengono emessi attraverso la console.
Un programma applicativo emette tutti i suoi risultati sulla console, se non viene "istruito" per farlo "altrove".
Alcune funzioni del sistema, relative alla sua amministrazione e alla manutenzione possono essere svolte solamente dalla console.

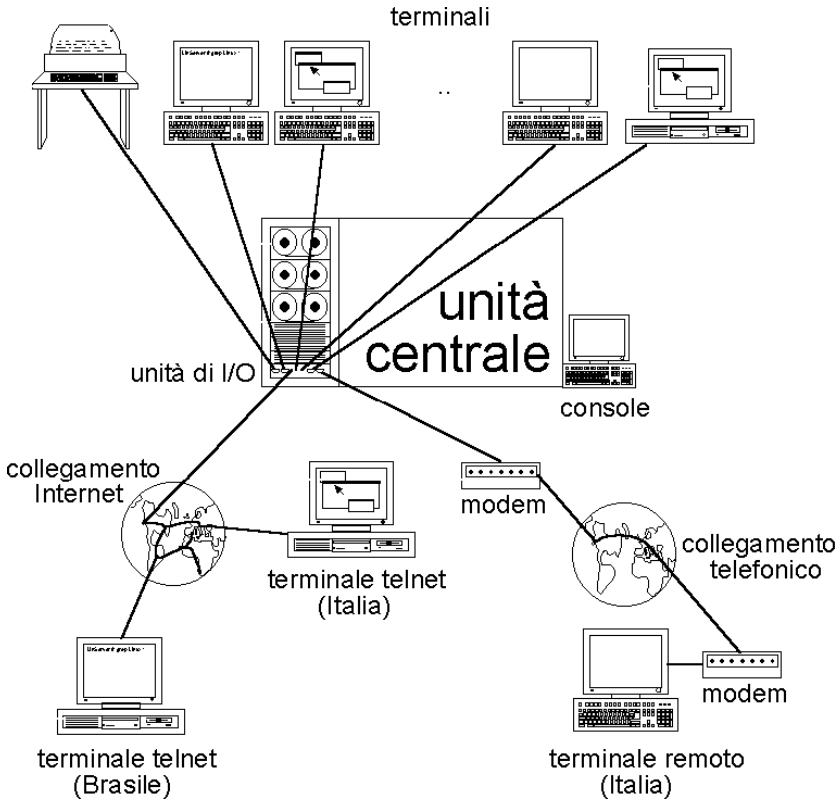


Figura 16: Sistema centralizzato

La console può essere collegata come un normale terminale, attraverso una porta seriale RS 232, oppure, più probabilmente, può fare uso diretto del monitor e della tastiera dell'unità centrale.

1.6 Livelli d'astrazione, macchine virtuali

L'approccio a macchina virtuale, del quale abbiamo accennato in precedenza, si può generalizzare, pensando ad un'architettura in cui siano presenti macchine virtuali incluse una dentro all'altra. Ogni macchina usa i servizi della sottostante e fornisce servizi più astratti a quella che le sta sopra. L'utente, all'esterno di tutti gli strati, può godere di tutti i servizi del S.O. In modo molto più semplificato.
Questo modo di vedere un S.O. viene di solito illustrato con diagrammi "a cipolla", come quello in Figura 17

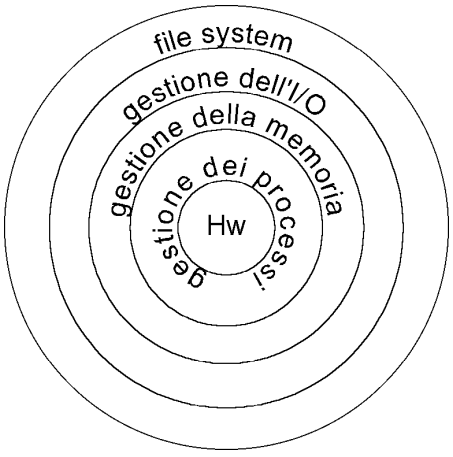


Figura 17 Stratificazione dei sistemi operativi

Il diagramma della figura è del tutto ideale, non fa riferimento a nessun S.O. esistente e serve solo per indicare quali potrebbero essere le risorse rese astratte ed un possibile ordine dal più fisico al più astratto.
I S.O. reali possono avere strati con un diverso ordine e, sopratutto, tendono a bypassare alcuni degli strati. Essi usano raramente una rigorosa architettura stratificata, sia per ragioni di efficienza, sia perché essi sono programmi così complicati che è molto difficile progettarli in modo "pulito"; inoltre durante la vita operativa di un S.O. esso subisce modifiche ed aggiunte di funzioni, che per motivi pratici tendono a deviare dall'architettura inizialmente progettata.

Lo strato più vicino all'hardware della Figura 17 ha il compito di realizzare il concetto di processo. Un utente che usi i servizi forniti da questo strato può creare ed utilizzare processi diversi, come se avesse a disposizione CPU diverse, una per ogni processo.
Lo strato di gestione dei processi deve mettere a disposizione funzioni che permettano la comunicazione fra i vari processi che girano contemporaneamente sul computer e metodi per la sincronizzazione delle loro attività.
Lo strato successivo indicato in figura gestisce la memoria. Esso realizza la memoria virtuale, la sua allocazione ai processi e deallocazione. Un programma che usi i servizi di questo strato può usare indirizzi di memoria virtuale, invece che indirizzi fisici.
Lo strato successivo (gestione dell'I/O) permette ai programmi che lo usano di avere una visione astratta dei dispositivi hardware. In alcuni S.O. (p.es. Unix) ogni dispositivo hardware deve essere visibile dall'esterno del S.O. come un semplice file. Questo strato si deve occupare di questo "mascheramento".
Nel disegno è presente anche lo strato "file system". Un file system si occupa della realizzazione del concetto di file. Al di sotto di questo strato le informazioni sui dischi rigidi sono accessibili solo come un insieme "sparpagliato" di informazioni, scritte sul supporto fisico in modo disordinato. Il **file** system organizza i settori dei dischi in file: unità logiche che contengono informazioni omogenee e che sono accessibili con un nome simbolico.
Il file sono a loro volta organizzati in strutture gerarchiche ad albero (directory), la cui gestione è anch'essa compito del file system.
Uno o più degli strati del disegno vengono raggruppati a formare il **kernel** [8]del Sistema Operativo.
Il kernel, o **nucleo**, è la parte del S.O. che sta a diretto contatto con l'hardware. Esso fornisce sempre le funzioni principali che riguardano i processi ed anche, ma non sempre, funzioni di astrazione dell'hardware e di memoria virtuale.

Spesso il file system viene realizzato al di fuori del kernel, questo permette di sostituirlo agevolmente ed anche di usare file system diversi sullo stesso computer. Se il file system è fuori dal kernel i relativi processi girano con privilegi d'utente e sono molto più sicuri per l'affidabilità generale del sistema.

Quando c'è un errore software in una procedura che fa parte del kernel c'è una buona probabilità che tutto il sistema vada in crash, mentre un errore in un programma d'utente di solito può venire recuperato con l'interruzione del solo programma che ha causato l'errore ed il mantenimento del controllo globale.
Il kernel è la parte del S.O. che esegue con i più alti livelli di priorità; al kernel "tutto è permesso".

Il kernel di un S.O. è un'entità che viene compilata congiuntamente. Esistono kernel molto piccoli ed altri molto voluminosi ..

L'architettura stratificata di un S.O. dà anche notevoli vantaggi per quel che riguarda la scrittura del software ed il relativo debugging .. .

API
Come già spiegato il S.O. nasconde il vero hardware del computer, presentando ai programmi applicativi ed all'utente una macchina virtuale che differisce dall'hardware reale ed è di utilizzazione più semplice.
Più in generale, quando uno strato di software nasconde ai suoi utilizzatori i suoi dettagli implementativi, permettendo una semplificazione nell'uso delle risorse del Sistema, quello strato si può chiamare "**risorsa virtuale**", o "oggetto astratto". Una risorsa virtuale permette ai suoi utilizzatori di operare in modo semplificato nell'uso di una risorsa "fisica" (o di più basso livello).
L'utilizzatore di una risorsa virtuale deve sapere come usarla. Le informazioni necessarie per poter utilizzare una risorsa virtuale si chiamano "**interfaccia**" della risorsa virtuale.
Per esempio, se la risorsa virtuale è un software, essa è costituita da un insieme di funzioni (procedure). Se conosciamo:
- il nome di ciascuna delle procedure della risorsa virtuale
- la sua funzione (cosa fa)
- il numero, il tipo, l'ordine e la funzione di ciascuno dei parametri delle procedure

siamo in grado di utilizzare la risorsa virtuale.
L'insieme delle informazioni appena citate va sotto il nome di "**interfaccia software**" o, più comunemente di **API** (**A**pplication **P**rogram **I**nterface).

Macchine virtuali a livello di linguaggio
Virtualizzazione di una CPU.
Es. Java VM, CLR del .NET framework.
Non c'entrano col S.O.

8 la parola "kernel" significa letteralmente "nocciolo" (della pesca) e dà una rappresentazione piuttosto calzante, considerando la Figura 17 :-)

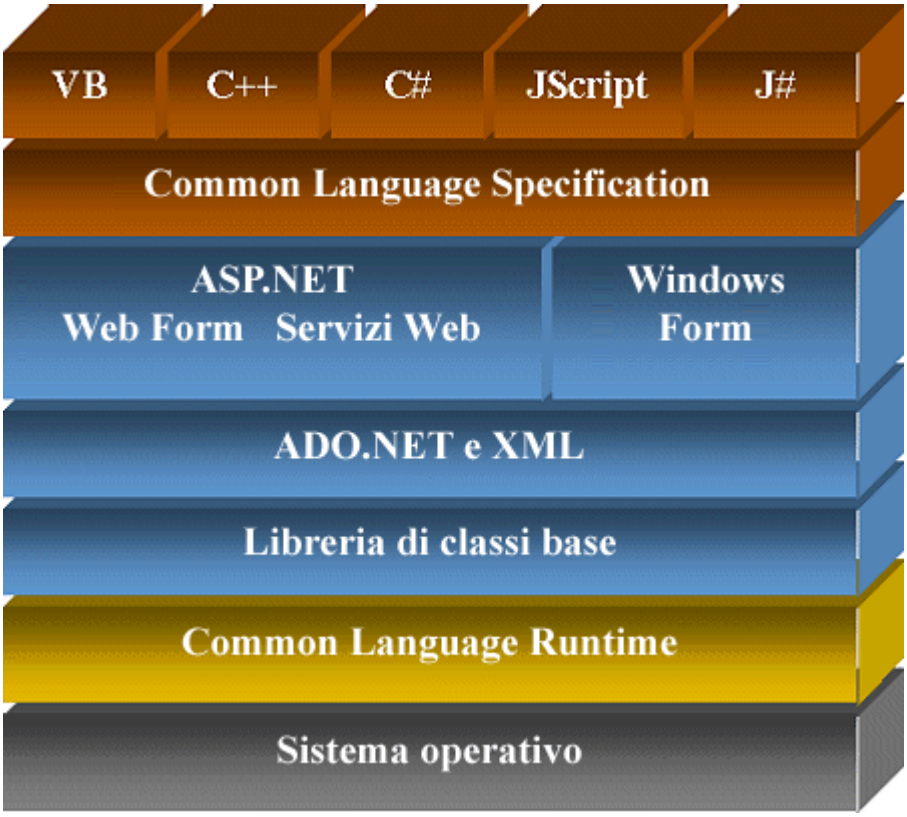


Figura 18 Stratificazione del software del framework .NET (da documentazione Microsoft)

Come si può vedere dalla figura si possono aggiungere strati di software ben oltre i confini del Sistema Operativo.

Virtual Machine Monitor (VMM)
Virtualizzazione di un computer.
Un VMM trasforma una singola macchina hardware in N istanze di computer virtuali indipendenti. Ciascuna di queste istanze è una "copia", non necessariamente identica, della macchina fisica, dotata di CPU con le stesse istruzioni della macchina fisica e di sue proprie risorse di sistema virtuali (memoria, I/O, periferiche). Non è detto che le risorse di sistema che vengono emulate siano identiche a quelle effettivamente presenti sulla macchina ospite.

Virtualizzazione ed emulazione
L'emulazione è una particolare forma di virtualizzazione. Si parla di emulazione quando un software che è stato creato per funzionare con un certo insieme di istruzioni viene fatto eseguire in un sistema che ha un altro insieme di istruzioni. Il software di emulazione tipicamente è un interprete, riceve le istruzioni con un set di istruzioni ed emette istruzioni nel set d'istruzioni della CPU fisica.

Periferiche

CPU

File system

Programmi di sistema ed utility
backup

1.6.1 S.O. "monolitici" o S.O. "modulari"

Se il kernel di un S.O. è costituito da un unico file, compilato e linkato :-(tutto insieme, si dice che esso è "monolitico". Dato che il kernel include anche i driver delle periferiche il fatto di avere un kernel monolitico non ne facilita l'aggiornamento, perché l'aggiunta di una nuova periferica significa la ricompilazione del kernel.
Le prime versioni del S.O. Linux avevano un kernel strettamente monolitico; invece oggi esso può caricare alcune sue parti, dette "moduli", a tempo di esecuzione. I driver dei dispositivi sono di solito usati sotto forma di moduli, per non doverli compilare tutti nel kernel, caricando invece solo quelli strettamente indispensabili.

Da questo punto di vista, analoga è la situazione di Windows.

1.6.2 S.O. a microkernel

Un S.O. a microkernel ha un kernel molto piccolo (micro :-).
Nel kernel di un sistema di questo tipo sono implementate le sole funzioni strettamente indispensabili, in alcuni casi solo la gestione dei processi e la parte più a basso livello della gestione dei dispositivi. Tenendo fuori dal kernel tutto il resto si aumenta la flessibilità del sistema, ed anche la sua sicurezza. Peraltro si può perdere in efficienza, dato che anche i programmi del S.O., per ogni servizio significativo, dovranno effettare chiamate al kernel, perdendo tempo.
!!!!to be completed !
Esempio: il kernel del S.O. In tempo reale "QNX" è 10 kByte, ha solo 14 chiamate. A livello 0 (ring 0) del processore girano solo il microkernel ed i servizi di risposta alle interruzioni. Tutti gli altri servizi (file system, gestione dei device, servizi di rete, interfaccia utente grafica (GUI) ..) girano isolati e protetti dal ring 0, come normali processi "utente". Esiste
un particolare processo detto "watchdog" il cui solo scopo è controllare che non ci siano violazioni di accesso alla memoria.

1.6.3 Altre classificazioni

S.O. "general purpose" o "dedicati"
La gran parte dei computer usa Sistemi Operativi generali, progettati per rispondere alle esigenze più diverse (S.O. "general purpose").
Peraltro alcune applicazioni "speciali" possono chiedere al Sistema Operativo prestazioni o funzioni che un S.O. "normale" non può assicurare. Dunque con certi particolari tipi di applicazioni non è possibile utilizzare Sistemi Operativi "general purpose" ma bisogna rivolgersi a S.O. specializzati.
Esempi di S.O. specializzati sono i S.O. realtime od i S.O. per la grafica (nelle console per giochi).

S.O. real-time
Ormai anche le apparecchiatura più impensate racchiudono una CPU; quando questo accade si usa parlare di dispositivi "intelligenti". Molti dispositivi "intelligenti" (detti anche dispositivi "**embedded**") hanno un software piuttosto semplice; in questi casi la CPU può eseguire un singolo programma che controlla tutto il funzionamento del dispositivo ed il suo "colloquio" con il mondo esterno.
Quando il software di un sistema embedded "cresce" fino a divenire moderatamente complesso è opportuno usare i servizi di un Sistema Operativo dedicato.
Le esigenze dei dispositivi embedded sono spesso molto diverse da quelle tipiche dei normali computer; esigenze analoghe sono poste anche da computer che devono intervenire su sistemi complessi, per assicurarne il monitoraggio od il controllo.
Il vincolo più importante posto a questi sistemi, che chiameremo "sistemi in **tempo reale**" (**realtime**), riguarda i tempi di risposta alle sollecitazioni che provengono dal mondo esterno.

Tempo reale
Quello di "tempo reale" è un concetto relativo.

Si può dire che un sistema è in tempo reale se fornisce i suoi risultati in tempo utile per lo scopo che ci si prefigge.

L'entità effettiva dell'intervallo di tempo massimo oltre il quale il sistema non è più in tempo reale dipende fortemente dall'applicazione, per esempio:
1. per il "rendering" grafico finale di un film animato si può attendere molti giorni, o diversi mesi
2. per il calcolo delle previsioni metereologiche a medio termine ci si può permettere di attendere diverse ore
3. per la regolazione di una temperatura il tempo fra le attuazioni può andare da pochi secondi a diversi minuti, in base alle dimensioni (capacità termica) dell'oggetto da regolare
4. per una transazione finanziaria (es.Bancomat) il tempo massimo per lo svolgimento dell'operazione è dell'ordine di una decina di secondi
5. per la realizzazione di movimenti meccanici rapidi (es. movimentazione automatica di materiale, bracci di robot) il tempo per la lettura dei sensori e l'esecuzione del programma che decide come comandare i motori può andare dai pochi millisecondi al secondo
6. per prendere decisioni in un dispositivo veloce di una rete di computer si deve impiegare da alcune decine di ns ad 1 µs (es. "switch" di rete locale o router (vedi il prossimo volume)).

Per rispondere ad esigenze così diverse in termini di tempo non si può scegliere sempre la stessa soluzione. Di solito per i casi 1, 2 e 4 vengono usati particolari combinazioni di computer "normali", con S.O. "general purpose", collegati in rete, o anche particolari sistemi con S.O. dedicati.
Per il caso 6 molto spesso non si può fare a meno di ricorrere al "puro hardware".
Negli altri casi si sceglie il S.O. "general purpose" se l'applicazione è semplice ed i tempi di risposta sono dell'ordine dei 10 ms, ma non devono essere rispettati strettamente, mentre bisogna pensare ad un S.O. realtime se si ha bisogno di ri-

sposte garantite entro pochi ms, oppure se il sistema è così complesso che un S.O. generale non garantirebbe risposte neppure nell'arco di tempo dei secondi.
Le esigenze più tipiche che portano alla scelta di un S.O. realtime sono le seguenti:
1. regolarità (determinismo) nei tempi di risposta,
2. velocità dell'I/O
3. accesso agli interrupt hardware
4. controllo delle priorità dei processi
5. possibilità di modifica delle "politiche" di gestione del sistema

Le prime due esigenze sono decisive ed esprimono l'approccio diverso con cui vengono progettati questi sistemi. Mentre i S.O. "general purpose" sono progettati per massimizzare l'efficienza nell'esecuzione dei programmi, quelli in tempo reale sono pensati per massimizzare le prestazioni in I/O. Per questo possono dare tempi di latenza "garantiti", per rispettare i quali sono disposti ad eseguire i programmi con una efficienza leggermente inferiore.
Alcune applicazioni tipiche per i sistemi in tempo reale sono:
- Supervisione e controllo di impianti chimici, industriali e reti di trasporti (reattori chimici e nucleari, linee di montaggio robotizzate, automazione ferroviaria, ..)
- Dispositivi "intelligenti" nell'automobile (ABS, airbag, controllo dell'accensione, ..)
- Domotica (automazione "della casa": decoder per pay-TV, antifurto, ..)

1.6.4 Memoria virtuale

Si parla di memoria virtuale quando i programmi possono utilizzare uno spazio di indirizzi molto più grande di quello fisicamente disponibile sul sistema su cui funzionano. Per poter realizzare questo obiettivo non si utilizza la sola memoria di lavoro (RAM), ma anche l'hard disk.
Ogni programma utilizza uno spazio di indirizzi molto ampio, detto memoria virtuale, e si "illude" di avere a disposizione una memoria più grande di quelle fisicamente disponibile.
Il sistema operativo e la CPU concorrono a far sì che tutta la memoria richiesta dai programmi sia disponibile, o nella memoria RAM, oppure sul disco fisso del sistema.
Sul disco fisso viene riservata un'area, detta "**area di swap**" ("swap area"), che viene destinata a contenere quelle parti della memoria virtuale che non trovano posto nella RAM.
Questa area viene gestita direttamente dal Sistema Operativo e può anche essere un vero e proprio file, nel qual caso viene chiamata "**swap file**".
Ad ogni accesso alla memoria fisica (p.es. per una MOV, in un programma applicativo), parte un meccanismo di trasformazione dell'indirizzo virtuale in un indirizzo fisico. Questo meccanismo è quasi sempre supportato da funzionalità hardware della CPU, che andrà automaticamente a cercare la corrispondenza fra indirizzo virtuale ed indirizzo fisico. Questa corrispondenza sarà trovata in opportune tabelle, situate in locazioni note della memoria. L'indirizzo fisico trovato in queste tabelle sarà effettivamente utilizzato per accedere alla RAM.
Naturalmente, dato che il numero delle locazioni di memoria virtuale è molto più grande di quello della memoria fisica, potrà darsi che la locazione virtuale non sia presente in memoria fisica. La CPU si accorgerà di questa mancanza consultando le tabelle ed innescherà un processo di lettura di un blocco di dati dal disco fisso.

Per far questo la CPU lancerà una particolare eccezione, mappata ad un ben determinato vettore d'interruzione. Le routine di risposta alle eccezioni fanno parte del Sistema Operativo, perciò, se manca la locazione di memoria virtuale voluta, verrà eseguita automaticamente una routine del S.O.. Se la memoria fisica non è usata completamente, il S.O. legge sull'hard disk il "blocco" che contiene l'indirizzo virtuale che non si era trovato e lo e scarica in memoria fisica.

Se la memoria fisica è completamente piena, il S.O. deciderà, con suoi criteri interni, quale parte della memoria fisica dovrà escludere, copiare sul disco rigido, e sovrascrivere con il "blocco" che contiene la locazione che mancava inizialmente, letto dal disco rigido. Questa operazione viene detta "**swapping**" (vedi).

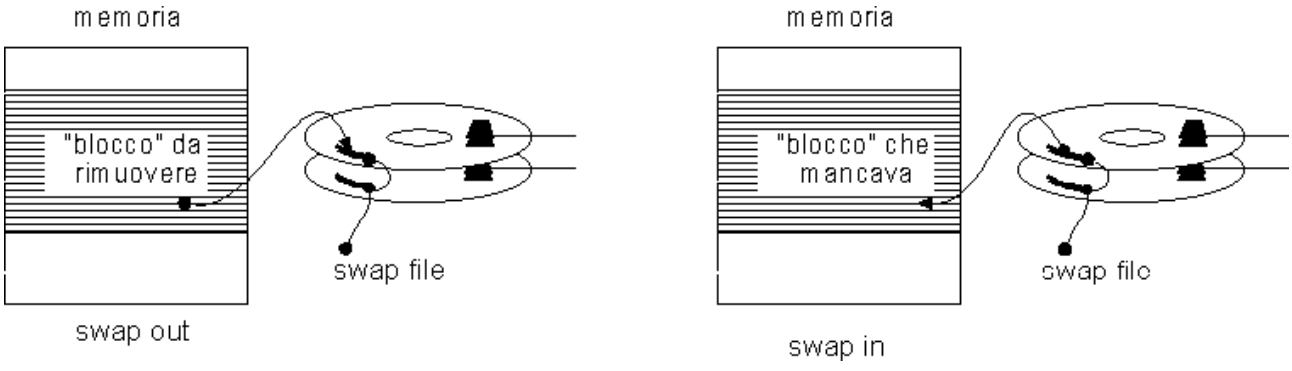


Figura 19: swapping

L'operazione di scrittura in memoria fisica di parte della memoria fisica "parcheggiata" sull'hard disk viene detta "**swap in**", mentre l'operazione contraria, dalla memoria all'hard disk, viene detta "**swap out**".
Il meccanismo che fornisce la memoria virtuale è trasparente al programma applicativo, che vede soltanto una memoria molto grande. Al software applicativo sono celati tutti i dettagli sull'implementazione effettiva della memoria virtuale.
È importante sottolineare che l'operazione di swapping è molto più lenta rispetto alle operazioni, anche complesse, che coinvolgono la memoria RAM, in quanto coinvolge la memoria di massa. Perché il computer sia veloce il S.O. dovrà fare tutto il possibile per rendere minime le operazioni di swapping, individuando le aree di memoria che sono usate raramente ed effettuando lo swap out su quelle.

Protezione
In un sistema multitasking si pone il problema di come proteggere il Sistema Operativo e gli altri task dalle istruzioni di quei task che, per errore o per attacco malizioso alla sicurezza del sistema, tentino di accedere a locazioni di memoria che non dovrebbero essere accessibili.
Per evitare che un task interferisca con gli altri si devono mettere in atto due meccanismi di **protezione**:
protezione della memoria
- evitare che processi non autorizzati possano accedere ad aree di memoria riservate

protezione delle istruzioni
- evitare che processi non autorizzati possano eseguire istruzioni pericolose per la sicurezza del sistema.

Protezione della memoria, istruzioni privilegiate (p.es. I/O, modifica dei vettori d'interruzione e del timer, CLI, STI)

Sicurezza nell'accesso ai file. Un sistema pensato per la multiutenza deve tenere traccia dei diritti di accesso per i directory ed i file.

Quando un S.O. applica la protezione esso deve essere in grado di distinguere diversi livelli di privilegio, e di assegnare ad ogni processo l'opportuno livello. Alcuni processi dovranno essere estremamente limitati nell'accesso alle risorse, mentre altri potranno avere la più ampia libertà. E' chiaro che il Sistema Operativo dovrà avere accesso ad ogni locazione di memoria ed utilizzare tutte le istruzioni della CPU, per cui dovrà avere il livello più privilegiato, mentre i normali programmi d'utente saranno confinati, per sicurezza, al livello meno privilegiato.
Il modo più semplice di suddividere i privilegi, realizzato anche nei S.O. più usati al giorno d'oggi, è distinguere fra due soli "livelli", uno destinato al S.O. e l'altro usato da tutti gli altri programmi.
Si dice che un processo è in "modo supervisore" (supervisory mode o **kernel – mode**) quando esegue con tutti i diritti sulla protezione, mentre è in "**user – mode**" quando non ha alcun privilegio particolare di protezione.
Alcuni Sistemi Operativi del passato hanno avuto diversi livelli di protezione, peraltro questo può comportare un tale aggravio nella gestione del sistema da renderlo poco efficiente. Per esempio, il S.O. Multics aveva 5 livelli di privilegio, ma essendo molto complesso ed essendo stato scritto in un'epoca in cui la CPU poteva dare scarso supporto hardware al S.O., era talmente lento ed ingombrante da non essere mai commercializzato.

Modi utente e supervisore

Accesso ai servizi offerti dal S.O.
Per avere accesso alle funzioni del S.O. il programma applicativo deve effettuare delle "chiamate al Sistema" (System calls, o "supervisory call"), che funzionalmente sono chiamate a procedure[9].

Multitasking "cooperativo" oppure "preemptive".
!!!!

Preemption
La preemption e l'interrupt del clock.
!!!!

Alcuni traducono to preempt con "prelazionare", termine che sinceramente è bruttino .. (in compenso "preempt" in Italiano suona proprio male!).

Politiche
In molti campi ed in molte occasioni il S.O. deve prendere delle decisioni. Una decisione implica sempre una scelta fra diverse alternative.
Il S.O. fa le sue scelte in base a precisi criteri, che in genere devono cercare il miglior compromesso fra prestazioni e uso di risorse.
L'insieme dei criteri che portano il S.O. a prendere decisioni in un determinato campo viene detta "**politica**" del S.O. in quel campo (**system policy**).

9 In alcuni S.O. per CPU X86 (MS-DOS, Linux) la chiamata avviene per mezzo di interrupt software, come già visto. In Windows la gran parte delle chiamate di sistema sono CALL a funzioni "esterne" al programma, che risiedono in librerie a collegamento dinamico (DLL: Dynamic Link Library)). Gli interrupt software, anche se utilizzano il sistema d'interruzione per il salto alla procedura, dal punto di vista della funzione che svolgono sono del tutto analoghi a chiamate a procedura.

1.6.5 Elaborazione parallela

Sistemi multiprocessore

SMP

Il sistema più semplice e diffuso di multiprocessing è **SMP** (**S**ymmetrical **M**ulti**p**rocessing). I sistemi SMP prevedono che le CPU del computer siano del tutto identiche fra loro e funzionino alla stessa frequenza di clock. In questo caso il software multiprocessore si semplifica drasticamente; per questo la maggior parte dei S.O. general purpose per computer multiprocessore funziona su sistemi SMP.
I sistemi SMP sono "simmetrici" perché sia la CPU che il software del S.O. sono "identici". Il S.O. non fa nessuna distinzione fra una o l'altra delle CPU del computer; esse possono eseguire processi qualsiasi, sia degli utenti che del S.O. stesso.
Il problema maggiore di SMP è la "**scalabilità**" ovverosia la difficoltà ad aumentare significativamente le prestazioni all'aumentare del numero delle CPU. In pratica, se aggiungere una CPU ad un sistema SMP con una sola CPU può portare ad un incremento, poniamo, dell'80% delle prestazioni; aggiungere una CPU ad un sistema SMP che ne ha già 5 porterebbe ad un aumento del 10%[10]. Il che significa che usare più di 4 o 5 CPU su un sistema SMP non è conveniente: il costo di un'ulteriore CPU non si "ripaga" con un aumento sufficiente delle prestazioni del sistema.

Parallelismo massivo

Se in un computer multiprocessore ci sono centinaia o migliaia di CPU ("parallelismo massivo") i sistemi SMP non sono più adatti e si apre la via ad una grande varietà di architetture, nelle quali di solito una o più CPU prendono il ruolo di "coordinatrici" delle altre e sono le uniche che possono far eseguire i processi del S.O..
I S.O. operativi per computer con architettura a parallelismo massivo di solito sono molto particolari, scritti apposta per un particolare tipo di applicazione o per uno specifico computer; si tratta dunque spesso di Sistemi Operativi "dedicati" (vedi oltre per la definizione).

Microprocessori multithreading e multicore

Nelle CPU per personal computer "desktop" degli ultimi anni è stata introdotta una funzionalità che sta in mezzo tra il monoprocessore ed il biprocessore, si tratta dell'"**hyperthreading**". Le CPU con architettura di questo tipo hanno sul chip una singola CPU la cui Execution Unit è in grado di operare con due istruzioni contemporaneamente. In questo modo, utilizzando in parallelo le unità di una singola CPU, possono essere eseguite contemporaneamente due istruzioni. La CPU emula il comportamento di due CPU e come tale appare all'esterno. Il computer che usa una CPU dualthreading crede di avere due CPU, anche se le prestazioni non saranno quelle di un "vero" sistema con due CPU.
Il passo successivo è stato quello di integrare sul chip del microprocessore due o più intere CPU ("core" di CPU) realizzando di fatto un vero sistema multiprocessore "on chip". A questa tecnologia è stato dato il nome "**multicore**".
Dunque un computer con una CPU dual core è un un vero e proprio sistema biprocessore di tipo SMP, che ha il vantaggio di richiedere meno spazio, consumo e di fare meno rumore di un sistema biprocessore con CPU separate. E' dunque ideale per i computer multiprocessore destinati all'uso personale ("desktop" computer). Per i potenti computer destinati a diventare server di rete di solito si preferisce usare microprocessori con CPU separate.

Sistemi distribuiti

Cluster

Un **cluster** è un insieme di computer che si presenta come un unico computer. I computer che fanno parte di un cluster hanno tutti lo stesso Sistema Operativo e sono collegati da una rete veloce.
La parola inglese "cluster", significando "grappolo" (dell'uva) dà bene l'idea di un insieme di nodi di elaborazione che si comporta come un "grappolo" unico.
Il S.O. per cluster, essendo uguale in tutti i computer del cluster, ha gli stessi descrittori di processo, per cui, con opportuni algoritmi di scheduling, può fare in modo che un processo che viene eseguito su un computer "migri" ad un altro, in base alle esigenze del momento. Se p.es. un computer è scarico di lavoro, lo notifica in rete a tutti gli altri computer del cluster; se un altro computer si trova invece con un carico di lavoro eccessivo può trasferire al computer scarico uno o più dei propri processi.
Alcuni S.O. per cluster possono funzionare con un processo che ha il codice in un computer e la memoria in un altro.
Il kernel del S.O. Linux è stato modificato da diversi team per poter funzionare in clustering (es. "Beowulf" o "OpenMosix").
I cluster di computer sono utilizzati per risolvere problemi "calcolo intensivi" quali

- Rendering di immagini digitali
  es. gli effetti speciali del film "Titanic" sono stati realizzati con un cluster di un centinaio di computer Linux Beowulf
- Calcolo tecnico e scientifico, previsioni del tempo, e finanziarie, indagini geologiche
  es. Sun usa cinque "compute farm" basate su cluster di workstation Sun per il calcolo delle simulazioni di progettazione dei circuiti integrati delle sue CPU. I cluster comprendono più di 600 workstation multiprocessore, con un totale di più di 4000 CPU, 3 TByte di memoria e 100 TByte di spazio in hard disk. Le CPU di questa

10 Queste figure percentuali non sono "esatte" perché non provengono da misurazioni, ma sono comunque verosimili.

"compute farm" lavorano con un tasso di utilizzazione del 97%. Analogo discorso per Intel che per le simulazioni usa cluster di computer con la potenza di 10000 workstation scientifiche.

Grid
!!!!

SETI @ Home

Bilanciamento del carico
!!!!

Curiosità
Nel 1981 un mainframe IBM 370 era un gigante che pesava diverse tonnellate, con 2 CPU, un MByte di memoria e decine di hard disk, ciascuno della dimensione di un frigorifero, che insieme potevano contenere un centinaio di MByte. Usava il Sistema Operativo MVS/370, che aveva un manuale di decine di volumi, era molto affidabile ed efficiente. Una macchina del genere, della potenza di un organizer tascabile di pochi anni fa, era in grado di collegare contemporaneamente, con prestazioni accettabili, i terminali di 2000 utenti.

Per la scrittura di Unix fu sviluppato un linguaggio ad alto livello, in questo modo fu più facile rendere il sistema portabile, dato che il ricorso all'Assembly fu davvero minimo. Il linguaggio sviluppato doveva peraltro essere in grado di eseguire istruzioni molto vicine alla macchina, dato che doveva servire per scrivere un Sistema Operativo.
Questo linguaggio di "livello intermedio", a metà fra l'Assembly e gli altri linguaggi di più alto livello, che ebbe anche più successo del S.O. per cui era stato inventato, ebbe la sua ispirazione da un misterioso linguaggio preesistente, che si chiamava "linguaggio B", e venne chiamato "linguaggio C" (C language).