

Sistemi Operativi

Lezione 1-2

Introduzione ai Sistemi Operativi

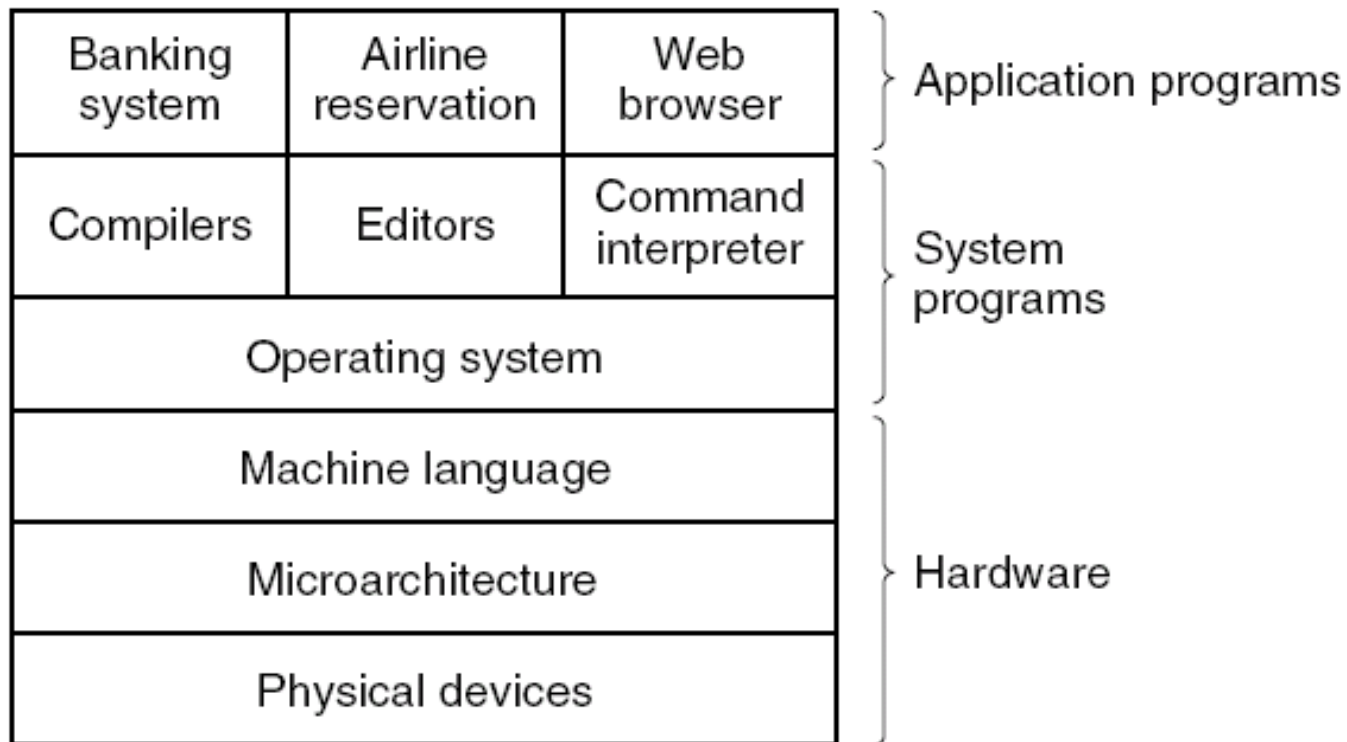
Il corso

- 4+3 ore di lezione settimanali (12 e 18 crediti)
- Le lezioni di laboratorio saranno interposte a quelle di teoria
- Esame:
 - Scritto con domande a risposta multipla + orale
 - Prova pratica per la parte di laboratorio
 - Sono previste 2 prove intermedie
- Libro di testo per la parte di teoria: Operating Systems Design and Implementation, di A. Tanenbaum e A. Woodhull, III ed.
- Sito ufficiale del corso: <http://homes.dico.unimi.it/sisop>

Il sistema operativo

- Il sistema operativo, è un insieme di programmi predisposti per assolvere ai seguenti compiti:
 - Gestire in modo ottimale le risorse di un calcolatore
 - Facilitare a programmatori ed utenti finali l'uso della sottostante macchina hardware

Una gerarchia di sistema



Kernel mode

- La caratteristica che contraddistingue il sistema operativo dalle altre componenti del software di base è la modalità d'esecuzione con cui opera
- Il sistema operativo, o più precisamente alcune sue componenti, è l'unico programma presente in un calcolatore che opera con il processore in **modalità kernel**

Il software di base

- Il SO è una componente del sw di base di un sistema
- I compiti che il software di base deve assolvere sono:
 - Abilitare l'uso del computer e delle sue componenti ad un utente
 - Gestire le risorse del sistema
 - Facilitare l'uso delle stesse ai programmatori di applicazioni
- Esempi di sw di base:
 - Compilatori e interpreti
 - DBMS
 - Sistemi operativi
 - Sistemi operativi di rete

Il software applicativo

- I compiti che il software applicativo deve assolvere sono:
 - Soddisfare le varie esigenze degli utenti finali (utilizzatori) in merito all'uso del calcolatore nelle loro attività
- Esempi di sw applicativo:
 - Word processor, foglio elettronico
 - Contabilità, Fatturazione
 - WWW, Posta elettronica, News

Evoluzione dei sistemi operativi

Le generazioni

- Generazione 1 (1945 – 55): Valvole modello open shop, applicazioni scientifiche
- Generazione 2 (1955 – 65): Transistor, sistemi batch, applicazioni scientifiche e prime applicazioni commerciali
- Generazione 3 (1965 – 80): ICs, sistemi multiprogrammati e timesharing, applicazioni commerciali
- Generazione 4 (1980 – Present): VLSI, applicazioni personali

Unità di misura

Exp.	Explicit	Prefix	Exp.	Explicit	Prefix
10^{-3}	0.001	milli	10^3	1,000	Kilo
10^{-6}	0.000001	micro	10^6	1,000,000	Mega
10^{-9}	0.000000001	nano	10^9	1,000,000,000	Giga
10^{-12}	0.0000000000001	pico	10^{12}	1,000,000,000,000	Tera
10^{-15}	0.0000000000000001	femto	10^{15}	1,000,000,000,000,000	Peta
10^{-18}	0.0000000000000000001	atto	10^{18}	1,000,000,000,000,000,000	Exa
10^{-21}	0.0000000000000000000001	zepto	10^{21}	1,000,000,000,000,000,000,000	Zetta
10^{-24}	0.000000000000000000000001	yocto	10^{24}	1,000,000,000,000,000,000,000,000	Yotta

Premesse

- **Tempo di turnaround:** intervallo di tempo che intercorre tra l'istante di tempo in cui si sottopone un processo (job) al sistema e l'istante di tempo in cui termina l'esecuzione del job
- **Execution time:** quantità di tempo utilizzata dalla CPU per eseguire un determinato processo
- **Utilizzo CPU in un intervallo di tempo T:** $\text{execution time}/T$
- **Troughput in un intervallo T:** numero di processi eseguiti in T

Premesse

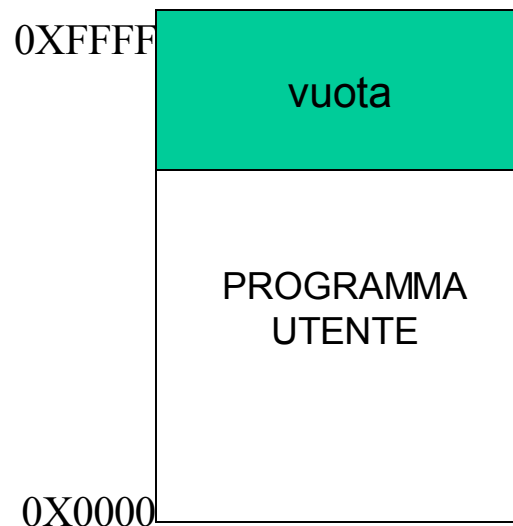
- Risorse usate da un'applicazione tipica
 - Input time manuale 15 min
 - Input time automatico 0.3 min
 - Output time 0.5 min
 - Execution time 1 min

Open Shop

- Gli utenti accedevano a turno al calcolatore dove caricavano ed eseguivano i loro programmi
- Sistema poco efficiente
 - Un utente ogni 20 min circa
 - Utilizzo CPU $1/20 = 5\%$

I Generazione

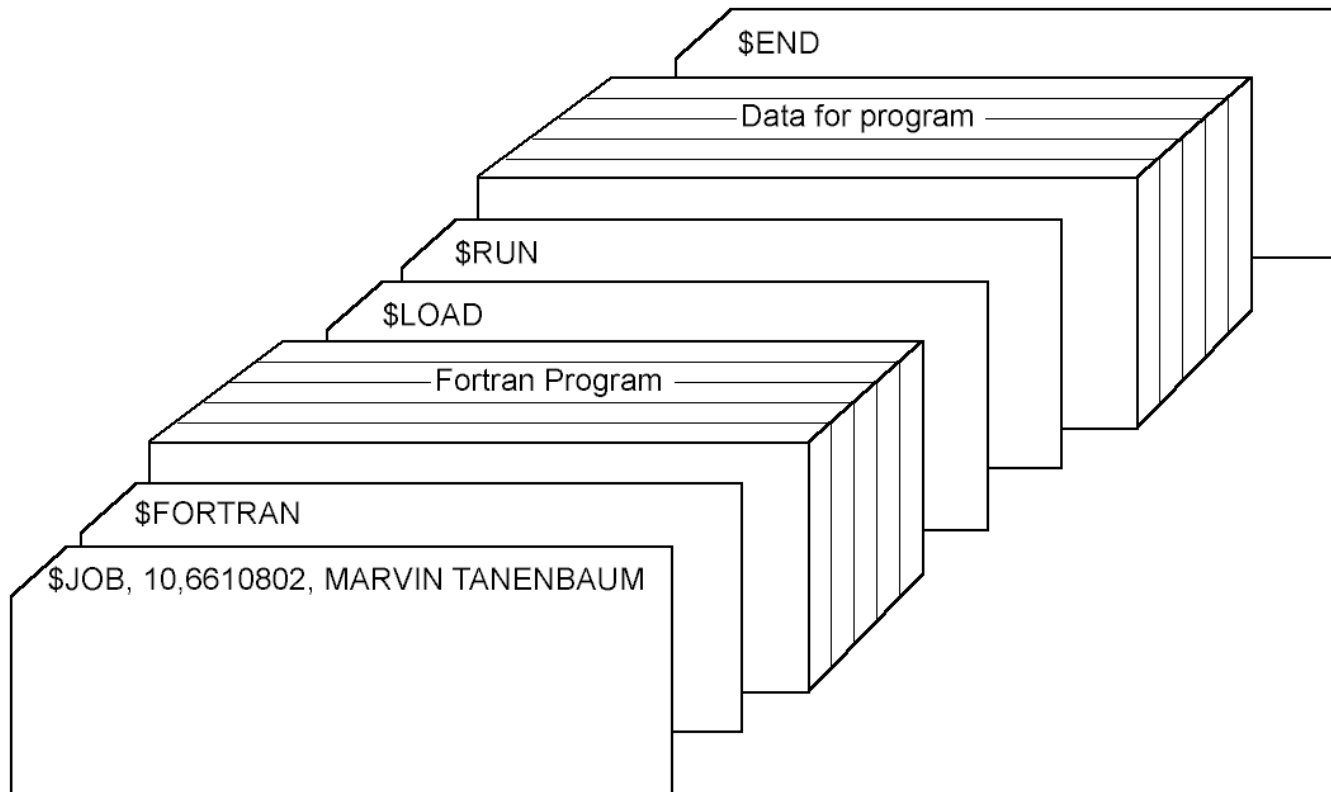
- In questo caso la memoria del calcolatore era completamente a disposizione dell'utente finale ed appariva in questo modo:



II Generazione

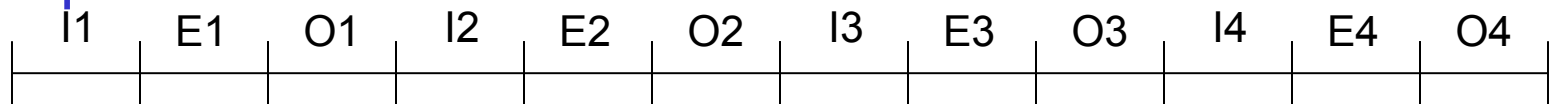
- Un primo miglioramento derivò dall'eliminazione dei tempi di input di dati e programmi con l'introduzione delle schede perforate
- L'utente preparava off-line il suo programma su schede perforate
- Accedeva al calcolatore a cui faceva leggere le schede, il programma veniva eseguito ed i risultati stampati

II Generazione



II Generazione

- Il profilo temporale dell'esecuzione dei programmi era quindi:



- In questo caso, se consideriamo il profilo di un'applicazione standard, il throughput del sistema diventa 33 job/h con un uso della CPU = $33/60 = 55\%$

II Generazione

- L'avvento dei lettori di schede incominciò a richiedere la presenza in memoria centrale di un programma che fosse in grado di:
 - Gestire l'unità periferica (CR)
 - Leggere le schede
 - Interpretare ed eseguire i comandi richiesti (Job Control Language)
 - Copiare il contenuto delle schede in memoria
 - A lettura terminata "lanciare" l'esecuzione del programma

II generazione

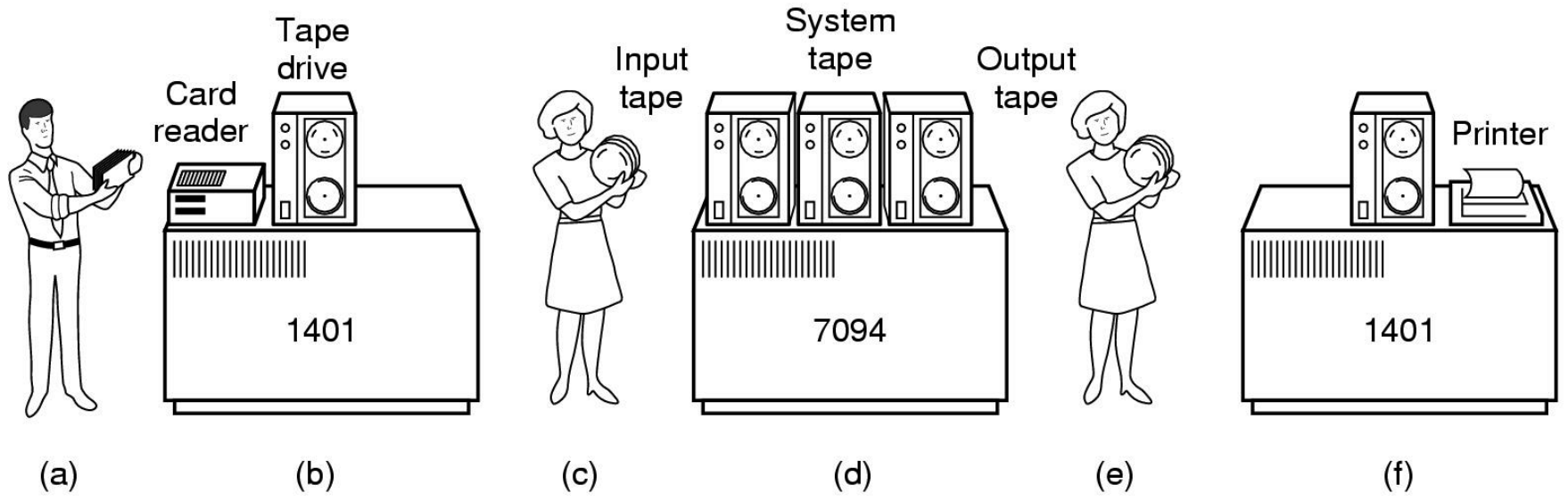
- In questo caso la memoria del calcolatore appariva in questo modo:

vuota
Programma utente
Assemblatore/ compilatore
Interprete JCL
Driver Card Reader
Driver Printer

Batch

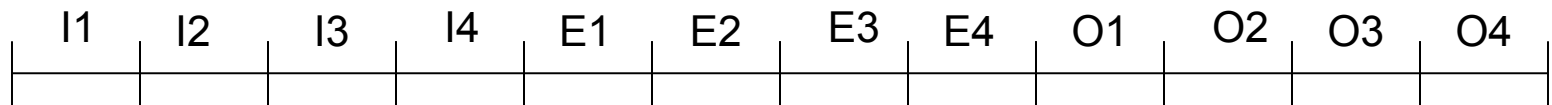
- Nasce l'esigenza di ottimizzare l'uso dei sistemi di calcolo e di eliminare dal "ciclo produttivo" le unità di I/O
- Le unità di I/O che fino a quel momento erano state collegate ai calcolatori, card reader e stampanti, vengono sostituite da unità a nastri molto più veloci
- Si modifica completamente lo schema di accesso ad un calcolatore

Batch



Batch

- Il profilo temporale diventa:



- In questo caso il throughput del sistema era di 55 job/h con un uso della CPU = $55/60 = 91\%$
- La CPU veniva utilizzata al meglio ma ...

Batch

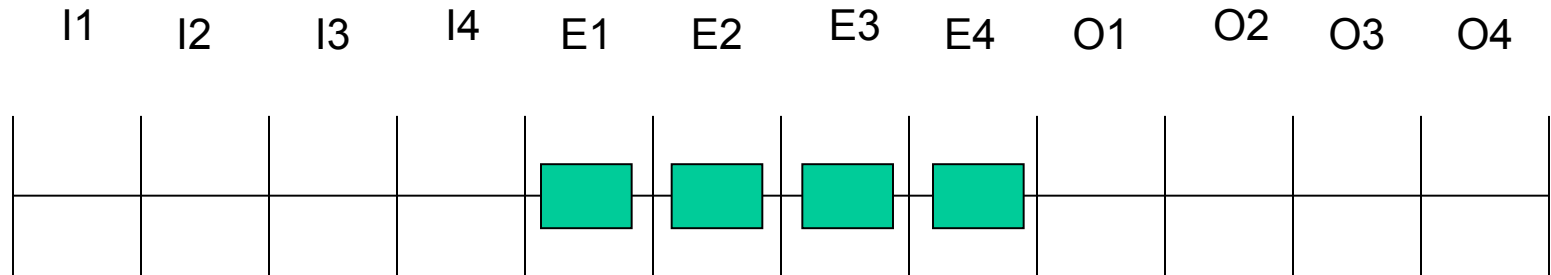
- Il tempo di risposta era diventato estremamente elevato
- L'utente doveva attendere la fine del batch per poter accedere ai risultati della propria elaborazione
- Nella maggioranza dei casi il tempo di risposta si aggirava intorno alle 24 ore

Batch

- Un ulteriore elemento di criticità presente nei sistemi batch era costituito dalla modifica delle applicazioni “tipo”
 - Sino alla fine degli anni '50 il calcolatore era usato principalmente per computazioni di tipo scientifico che richiedevano quindi un uso intensivo della CPU (CPU bound)
 - Dalla fine degli anni '50 il calcolatore incominciò ad essere utilizzato in applicazioni commerciali, caratterizzate da un forte uso di I/O

Batch

- Si assisteva quindi al seguente fenomeno:



tempo dedicato a gestire le periferiche

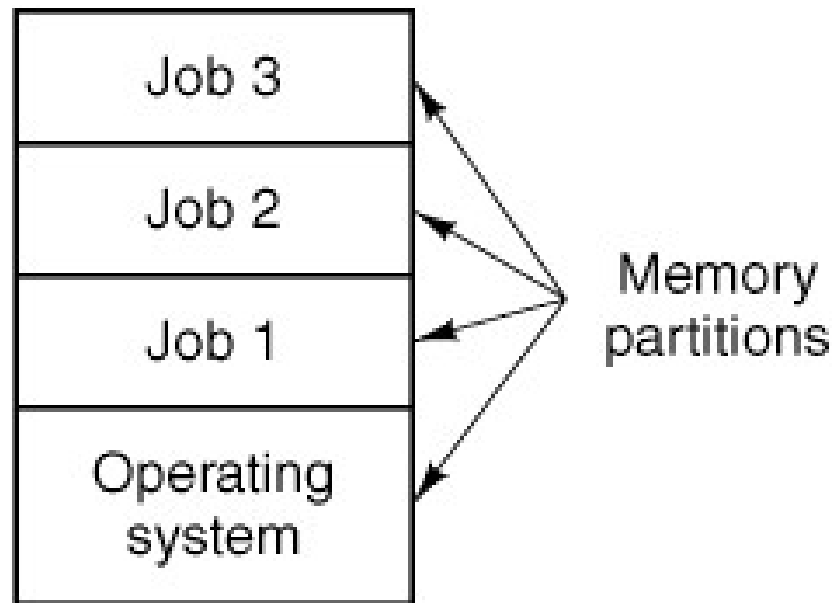
III Generazione: Multiprogrammazione

- Applicazioni tipicamente commerciali o molto interattive
- Nascono come risposta al problema di un miglior sfruttamento della CPU
- SO di riferimento: IBM OS/360

Multiprogrammazione

- I sistemi multiprogrammati godono delle seguenti caratteristiche:
 - Consentono la coesistenza CONTEMPORANEA in memoria centrale di due o più programmi
 - Adottano meccanismi che consentono di svincolare l'attività della CPU da quella delle periferiche di I/O (vedi slide successive)
 - Ottimizzano l'uso della CPU

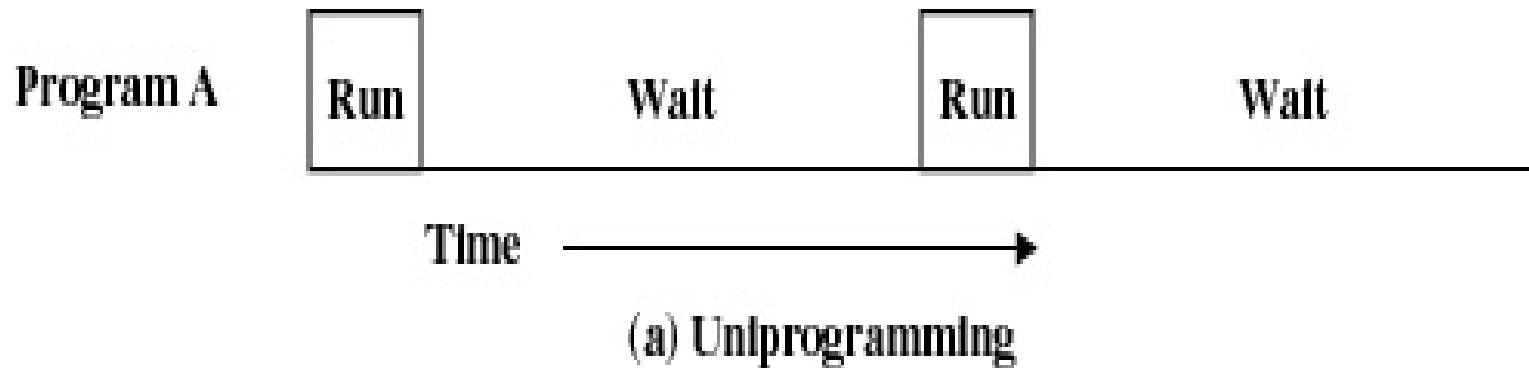
Multiprogrammazione



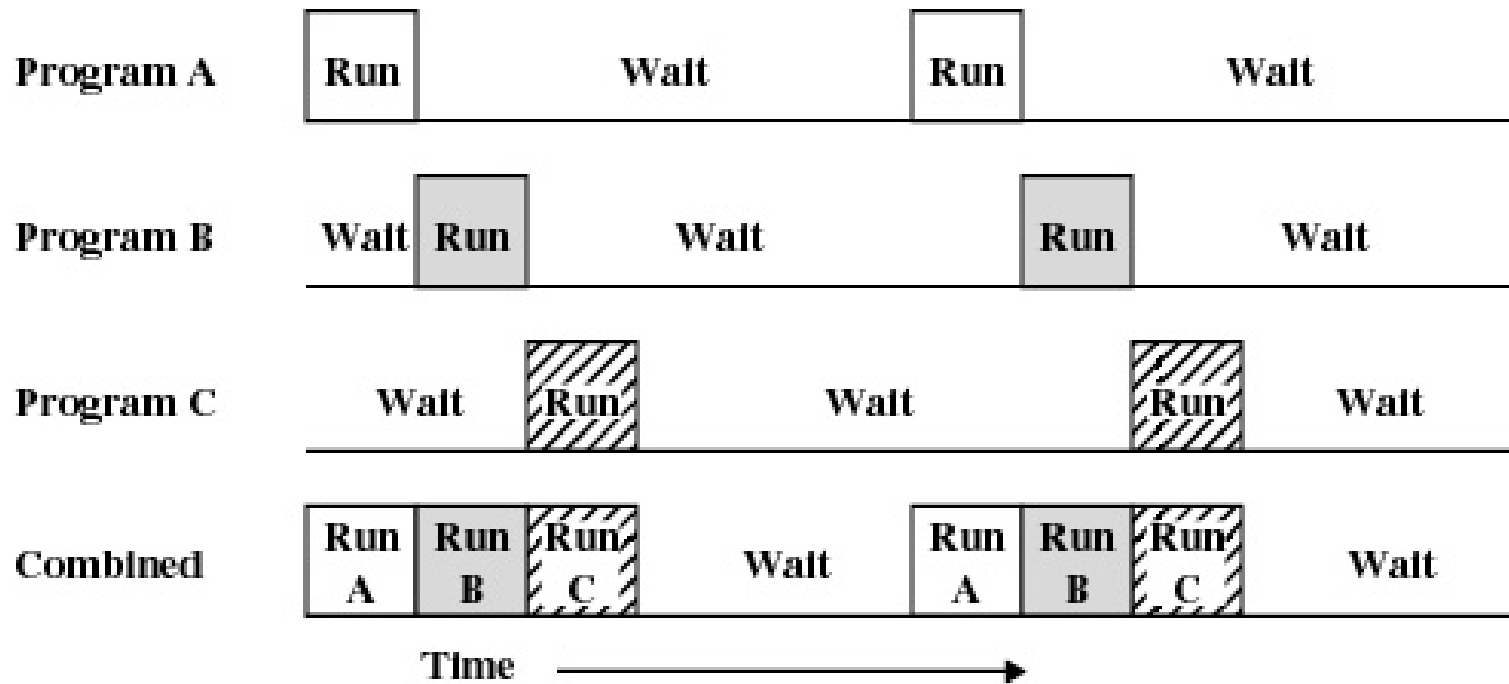
Multiprogrammazione

- Principio di funzionamento:
 - Viene scelto un programma P1 da eseguire, tra quelli presenti in memoria
 - Quando P1 richiede lo svolgimento di un'operazione di I/O, la stessa viene demandata alla periferica, il programma P1 viene momentaneamente sospeso, sarà ripreso successivamente
 - La CPU procede nell'esecuzione di un altro programma

Mono programmazione



Multiprogrammazione



(c) Multiprogramming with three programs

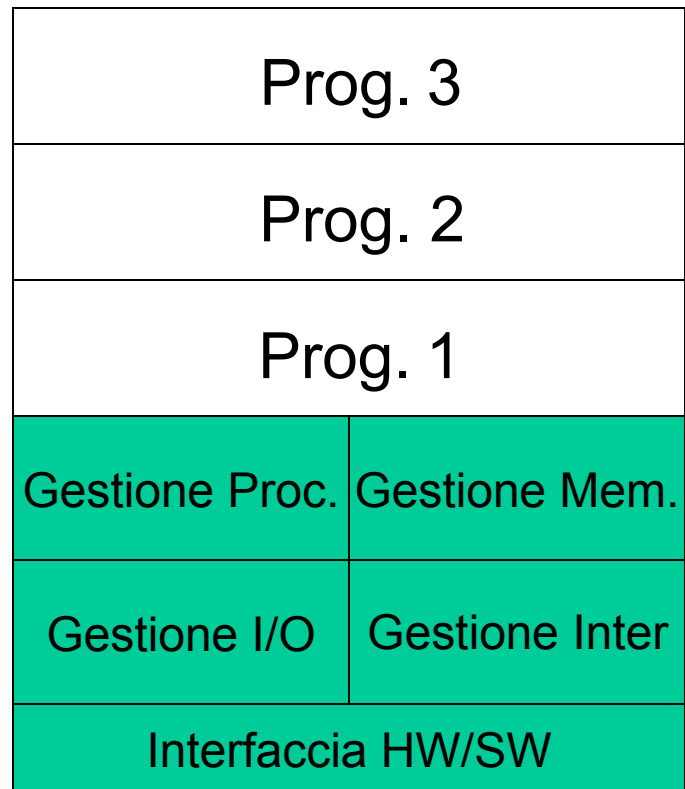
Interrupt

- Per realizzare questa modalità ci si è rifatti ad un uso intensivo dei segnali di interrupt
- L'interrupt è un segnale elettrico inviato da un dispositivo esterno, al microprocessore più precisamente al controller
- L'interrupt consente al processore di interrompere le attività in corso e di eseguirne altre

III Gen.: Multiprogrammazione

- Affinché lo schema illustrato funzioni, è necessario disporre di:
 - Routine di gestione degli interrupt
 - Moduli per la gestione dei programmi sospesi e di quelli pronti all'esecuzione
 - Moduli per la gestione delle periferiche
 - più processi possono richiedere l'uso della stessa risorsa
 - Moduli per la gestione della memoria
- Tutte queste funzionalità sono accorpate nel sistema operativo

III Gen.: Multiprogrammazione



Time Sharing

- Sistemi altamente interattivi
- Ogni utente lavora interattivamente e riceve periodicamente attenzione dal calcolatore
- L'impressione che ne ottiene è di avere a disposizione un sistema dedicato
- Un sistema time-sharing non deve necessariamente essere multiprogrammato
- CTS5 → Multics → Unix

Batch vs. Time sharing

	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

Real-time

- Sistemi che devono garantire l'esecuzione di certe operazioni entro un limite di tempo prefissato
- La nozione di real-time è legata all'ambito applicativo
- Sistemi hard real-time
 - i vincoli di tempo non possono essere violati pena gravi danni al sistema e all'ambiente
- Sistemi soft real-time
 - il mancato rispetto dei vincoli di tempo porta ad un degrado delle prestazioni del sistema

IV Generazione

- Sistemi operativi più facili da utilizzare
 - Installabili da utenti poco esperti
 - Utilizzabili da ogni classe di utente
- Nasce la nozione di sistema **user-friendly**
 - Molta enfasi viene posta sulle interfacce utente, grazie soprattutto al lavoro della Apple che introduce le interfacce a finestre

IV Generazione

- I principali sistemi operativi di IV generazione sono:
 - UNIX
 - molto diffuso su workstation nelle varie versioni: HP-UX, SOLARIS, AIX, ULTRIX
 - in ambiente PC: LINUX, FreeBSD, OpenBSD
 - MS-DOS → W95 → WNT → W2000 → XP → VISTA
 - Nato per PC IBM e compatibili, che usavano il processore Intel 8088
 - Con WNT Microsoft ha iniziato ad operare sul mercato delle WKS

IV Generazione

- I sistemi operativi di IV generazione hanno dovuto, per primi, fare i conti con una nuova risorsa: **LA RETE**
- A partire dalla metà degli anni '80, lo sviluppo di protocolli per reti locali prima e per reti geografiche immediatamente dopo ha favorito lo sviluppo delle reti di calcolatori

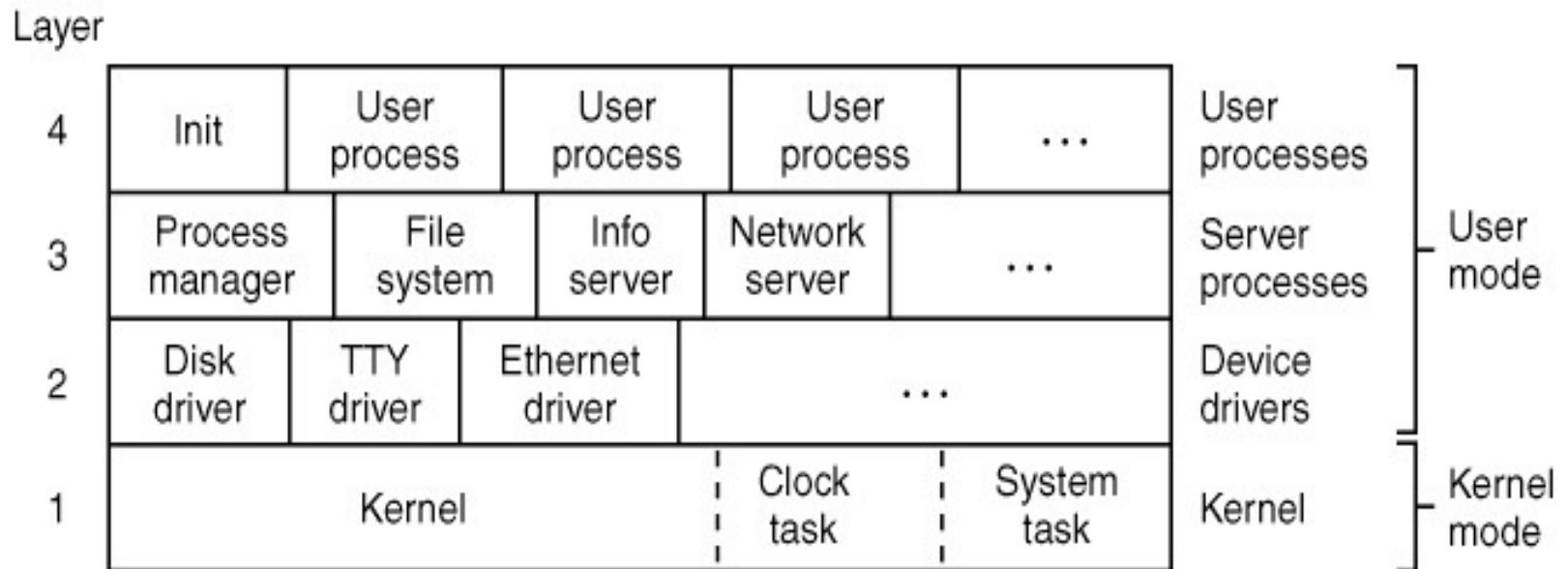
IV Generazione

- Per poter operare in una rete, un calcolatore deve essere predisposto del necessario
 - HW
 - cavi e schede di rete
 - SW
 - protocolli per la comunicazione con gli altri host collegati in rete
- I protocolli più diffusi
 - Ethernet per le reti locali
 - TCP-UDP/IP per le reti geografiche

IV Generazione

- Si assiste ad una continua evoluzione verso l'usabilità del sistema, che richiede interfacce sempre più vicine all'uomo
- Vi è una continua evoluzione di unità periferiche e sistemi di interconnessione (bluetooth, wi.fi, digital camera, biometric devices) che il sistema operativo deve essere in grado di inglobare e gestire
- La complessità del So sta crescendo

Architettura di un SO: minix



Architettura di un SO: W2000

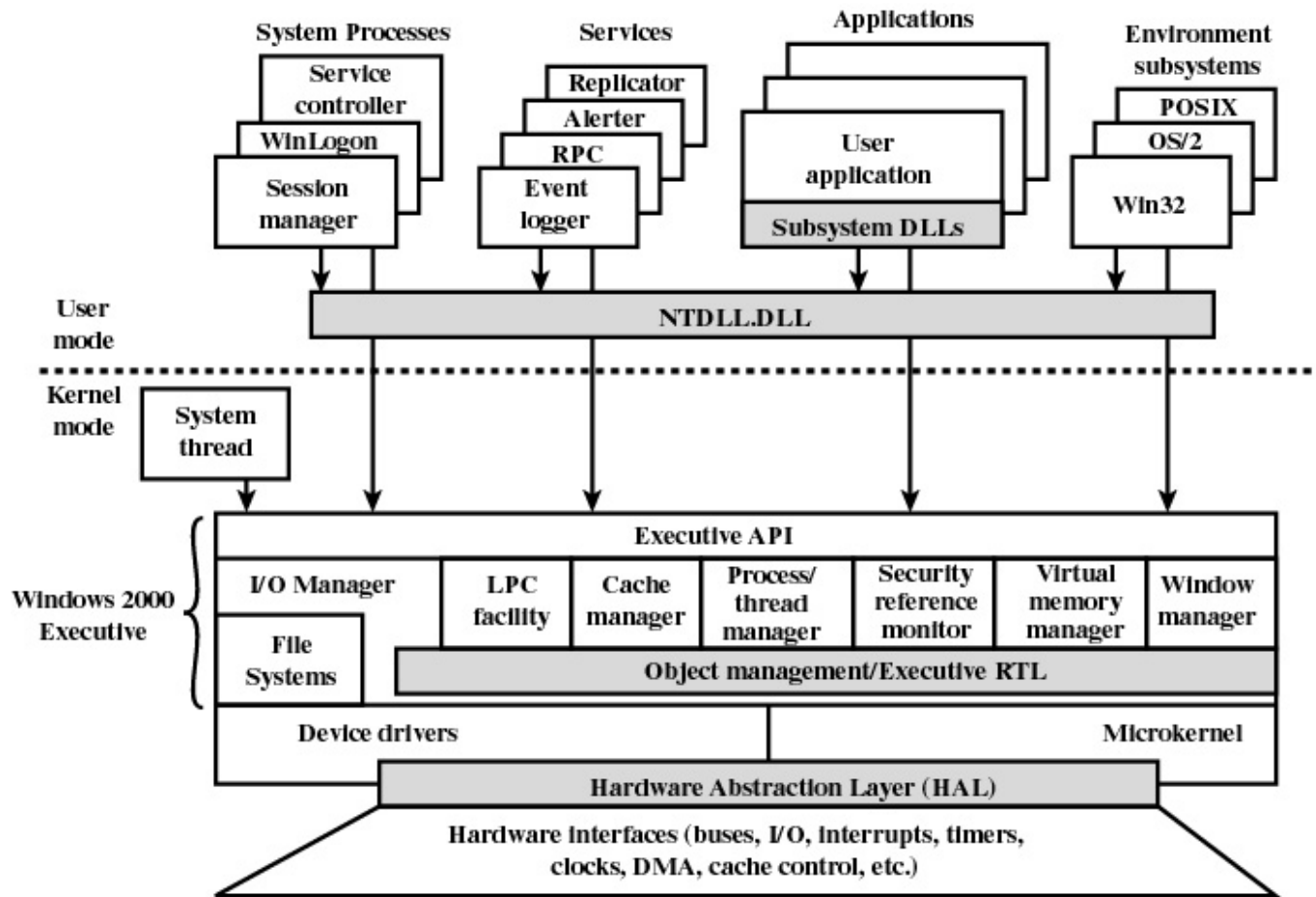


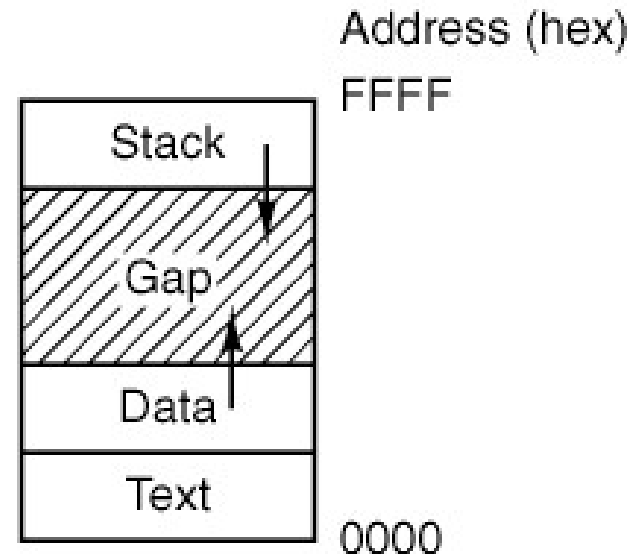
Figure 2.13 Windows 2000 Architecture

Le funzionalità di un SO

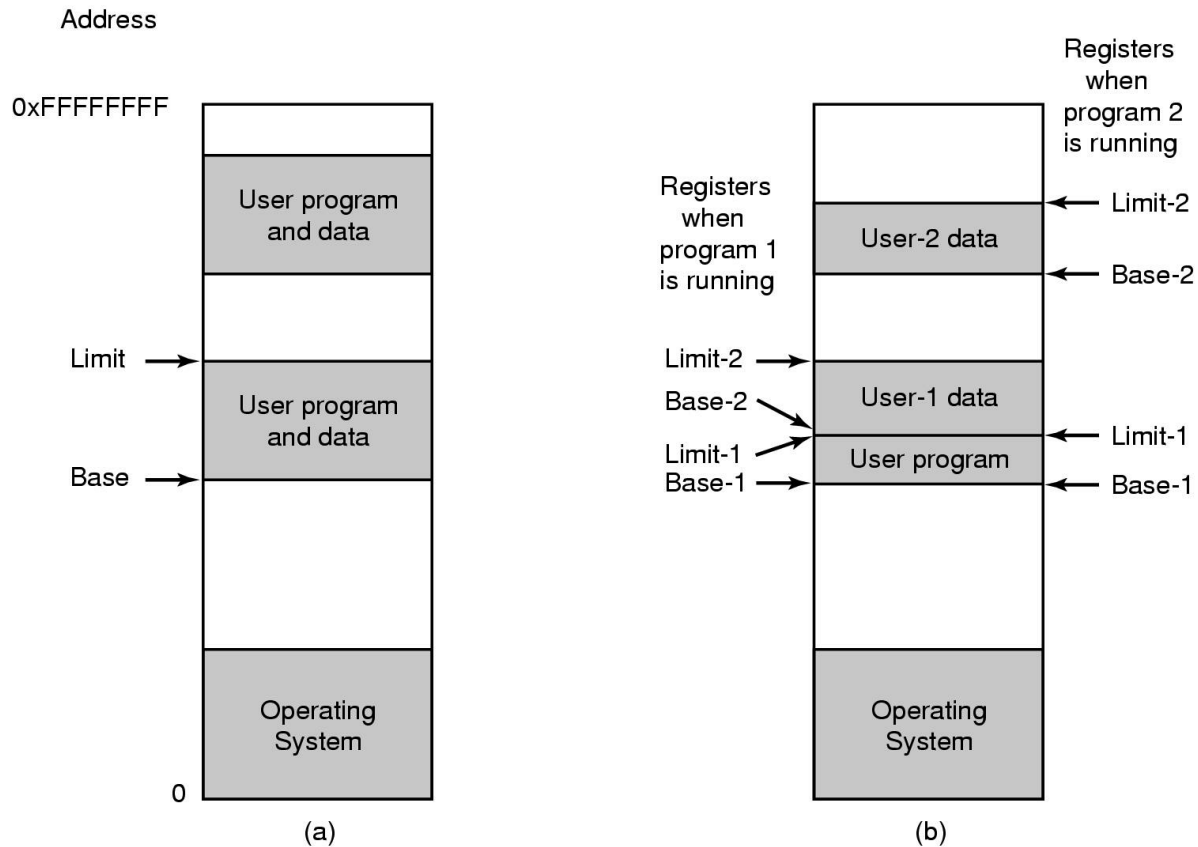
- L'insieme dei programmi che compongono un SO svolgono le seguenti funzioni:
 - Inizializzazione del sistema
 - Gestione dei processi
 - Gestione della Memoria
 - Gestione delle periferiche I/O
 - Gestione apparati di comunicazione
 - Gestione dei file
 - Gestione Syscall/interrupt
 - Protezione
 - System Utility

Processi

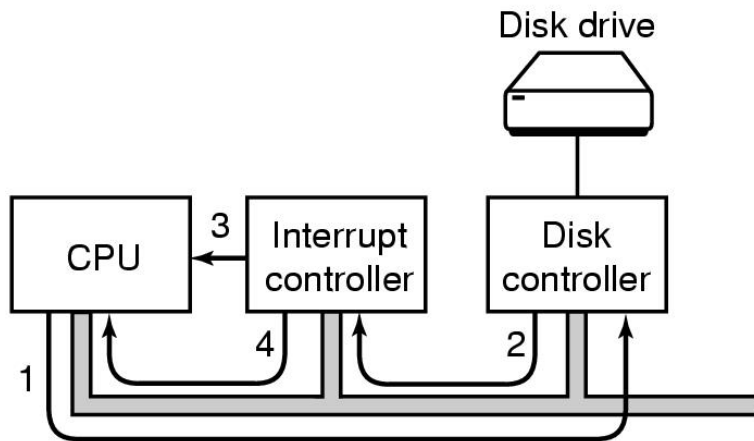
- Un processo è una qualunque attività svolta dal sistema operativo ed è sempre riconducibile all'esecuzione di un programma
- Un processo è caratterizzato da tre componenti: data, testo, stack



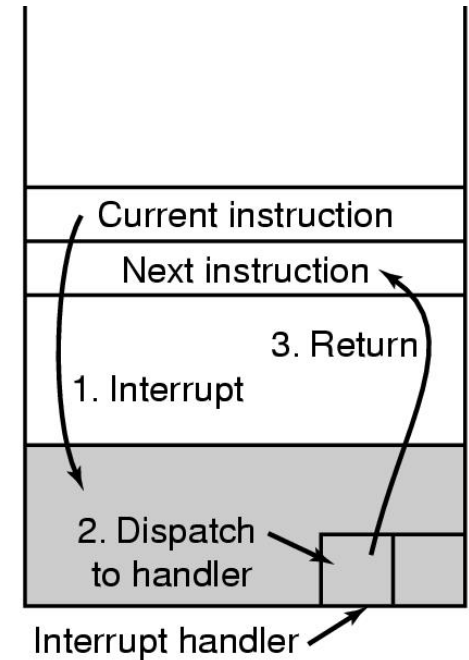
Gestione della Memoria



Gestione I/O e comunicazioni



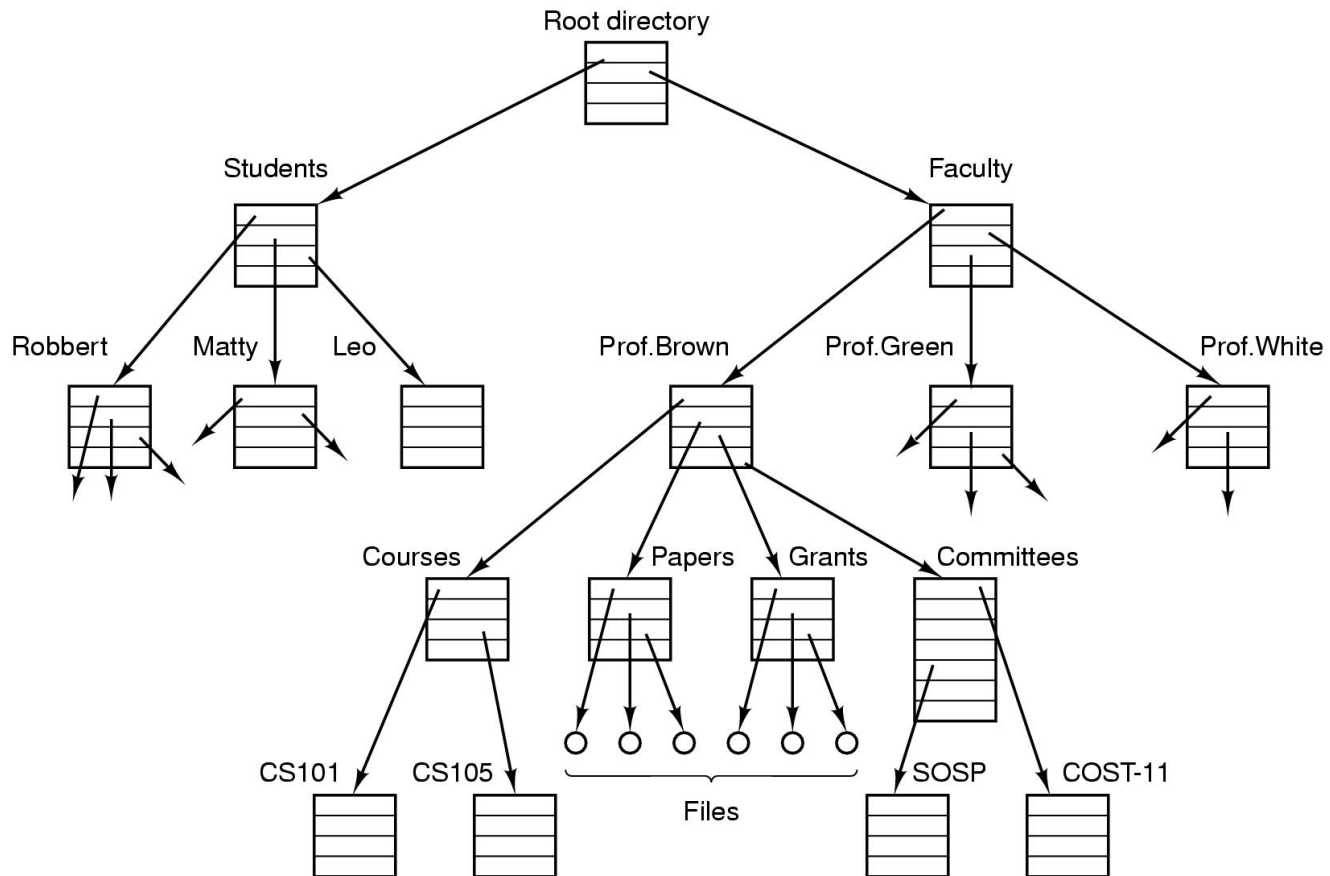
(a)



(b)

Avvia, interrompe e controlla il flusso di dati da e verso i dispositivi di I/O

File system



Syscall

- Il sistema operativo è **gestore esclusivo** di tutte le risorse di un sistema al fine di:
 - Garantire una gestione ottimale delle stesse
 - Evitare collisioni
 - Fornire modalità semplificate per il loro uso
- ... quindi, chiunque debba usare una risorsa deve rivolgersi al sistema operativo:
 - Come?

I servizi

- Il meccanismo usato per questo tipo di comunicazioni sono le **SYSTEM CALL** (o syscall) (Win32 API in Windows)
- Le syscall sono delle chiamate a procedure di sistema che svolgono particolari funzioni
- Le syscall sono quindi l'interfaccia del sistema operativo verso le applicazioni
- Il formato delle syscall differisce tra le varie implementazioni dei sistemi operativi
- In ambito UNIX è stato fatto uno sforzo di standardizzazione: POSIX

Syscall

- Le syscall sono divise per funzionalità
 - Process Management
 - Signal
 - File Management
 - Directory and file system management
 - Protection
 - Time management
 - I/O

System Call

Process Management

<code>pid = fork()</code>	Create a child process identical to the parent
<code>pid = waitpid(pid, &statloc, opts)</code>	Wait for a child to terminate
<code>s = wait(&status)</code>	Old version of <code>waitpid</code>
<code>s = execve(name, argv, envp)</code>	Replace a process core image
<code>exit(status)</code>	Terminate process execution and return status
<code>size = brk(addr)</code>	Set the size of the data segment
<code>pid = getpid()</code>	Return the caller's process id
<code>pid = getpgrp()</code>	Return the id of the caller's process group
<code>pid = setsid()</code>	Create a new session and return its process group id
<code>l = ptrace(req, pid, addr, data)</code>	Used for debugging

Esempio

```
#define TRUE 1

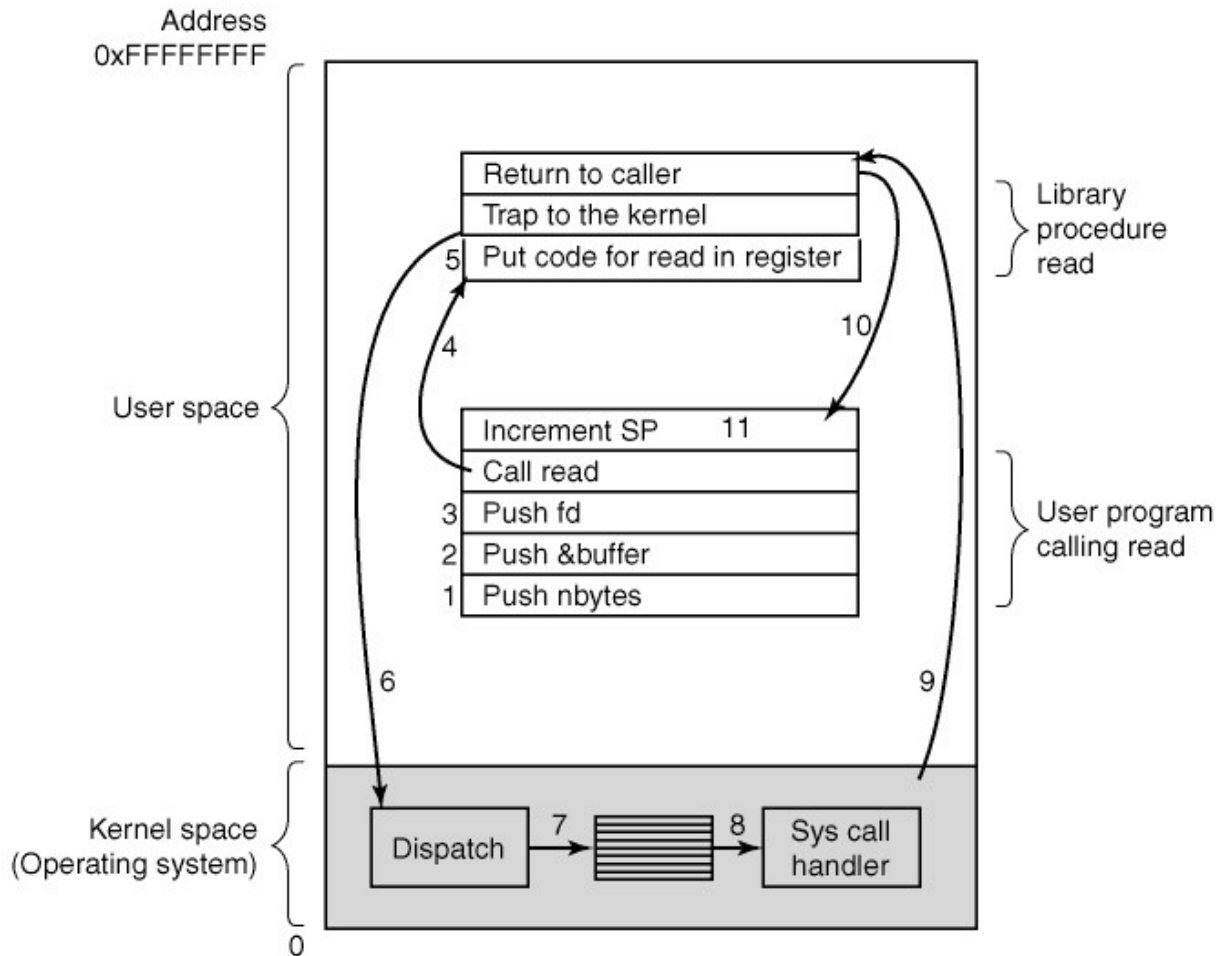
while (TRUE) {
    type_prompt( );
    read_command(command, parameters);

    if (fork() != 0) {
        /* Parent code. */
        waitpid(-1, &status, 0);
    } else {
        /* Child code. */
        execve(command, parameters, 0);
    }
}
```

Syscall

- Le componenti principali di una syscall, in ambito POSIX, sono
 - Una libreria di chiamate a syscall
 - La libreria di syscall
- Un programma applicativo che vuole accedere ad una syscall deve solitamente:
 - Chiamare la routine di libreria collegata
 - Questa provvede a formattare i parametri opportunamente e a richiamare la syscall effettiva

Chiamata di una Syscall



System Call

File Management

<code>fd = creat(name, mode)</code>	Obsolete way to create a new file
<code>fd = mknod(name, mode, addr)</code>	Create a regular, special, or directory i-node
<code>fd = open(file, how, ...)</code>	Open a file for reading, writing or both
<code>s = close(fd)</code>	Close an open file
<code>n = read(fd, buffer, nbytes)</code>	Read data from a file into a buffer
<code>n = write(fd, buffer, nbytes)</code>	Write data from a buffer into a file
<code>pos = lseek(fd, offset, whence)</code>	Move the file pointer
<code>s = stat(name, &buf)</code>	Get a file's status information
<code>s = fstat(fd, &buf)</code>	Get a file's status information
<code>fd = dup(fd)</code>	Allocate a new file descriptor for an open file
<code>s = pipe(&fd[0])</code>	Create a pipe
<code>s = ioctl(fd, request, argp)</code>	Perform special operations on a file
<code>s = access(name, amode)</code>	Check a file's accessibility
<code>s = rename(old, new)</code>	Give a file a new name
<code>s = fcntl(fd, cmd, ...)</code>	File locking and other operations

System Call

Dir. & File System Mgmt.

<code>s = mkdir(name, mode)</code>	Create a new directory
<code>s = rmdir(name)</code>	Remove an empty directory
<code>s = link(name1, name2)</code>	Create a new entry, name2, pointing to name1
<code>s = unlink(name)</code>	Remove a directory entry
<code>s = mount(special, name, flag)</code>	Mount a file system
<code>s = umount(special)</code>	Unmount a file system
<code>s = sync()</code>	Flush all cached blocks to the disk
<code>s = chdir(dirname)</code>	Change the working directory
<code>s = chroot(dirname)</code>	Change the root directory

System Call

Protection

`s = chmod(name, mode)`

`uid = getuid()`

`gid = getgid()`

`s = setuid(uid)`

`s = setgid(gid)`

`s = chown(name, owner, group)`

`oldmask = umask(complmode)`

Change a file's protection bits

Get the caller's uid

Get the caller's gid

Set the caller's uid

Set the caller's gid

Change a file's owner and group

Change the mode mask

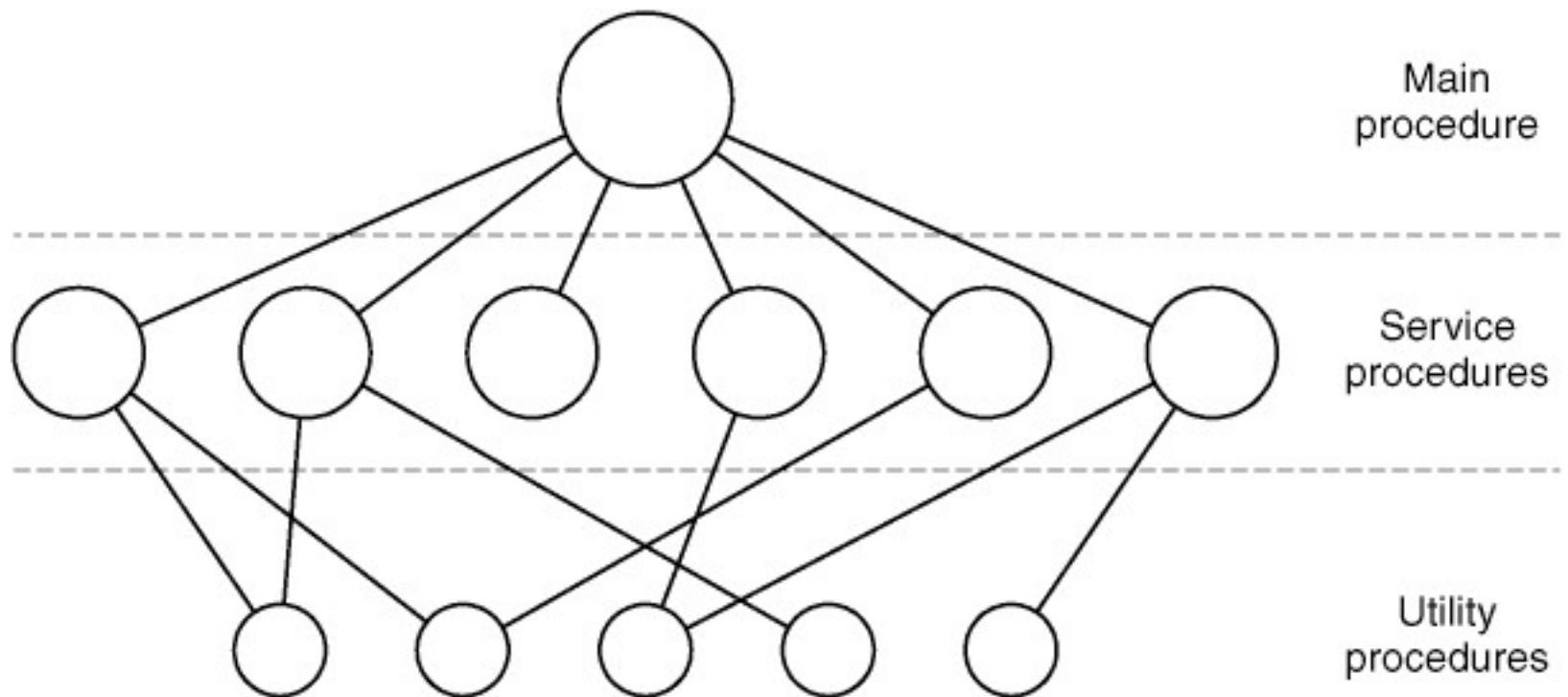
Shell

- Ogni utente di un computer può interagire direttamente con il sottostante SO fornendogli dei **comandi** da eseguire (crea un file, esegui un programma, ecc.)
- Questi comandi, prima di essere eseguiti devono essere opportunamente interpretati
- Il programma che interpreta i comandi, non è propriamente una componente del SO opera infatti in user-mode e si chiama **SHELL**

Struttura di un SO

1. Un programma principale che richiama sottoforma di procedure la parti di programma che implementano i diversi servizi del SO
2. Un insieme di procedure che implementano e realizzano le diverse chiamate di sistema (syscall)
3. Un insieme di procedure che svolgono una serie di compiti comuni alle diverse chiamate di sistema (abilita interrupt, disabilita interrupt, inizializza registri, ecc. ecc.)

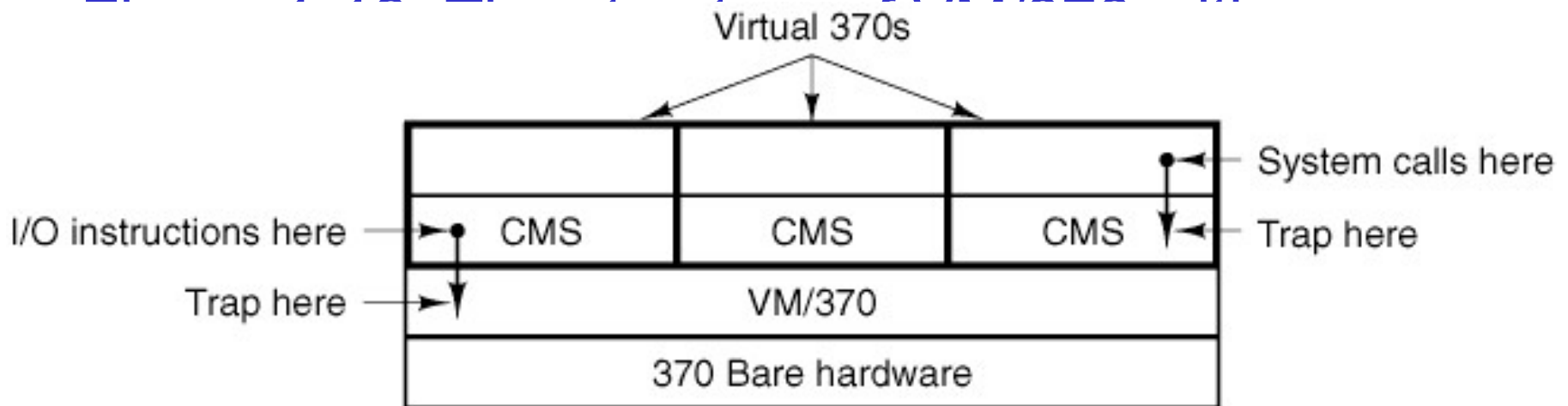
Layered System (1)



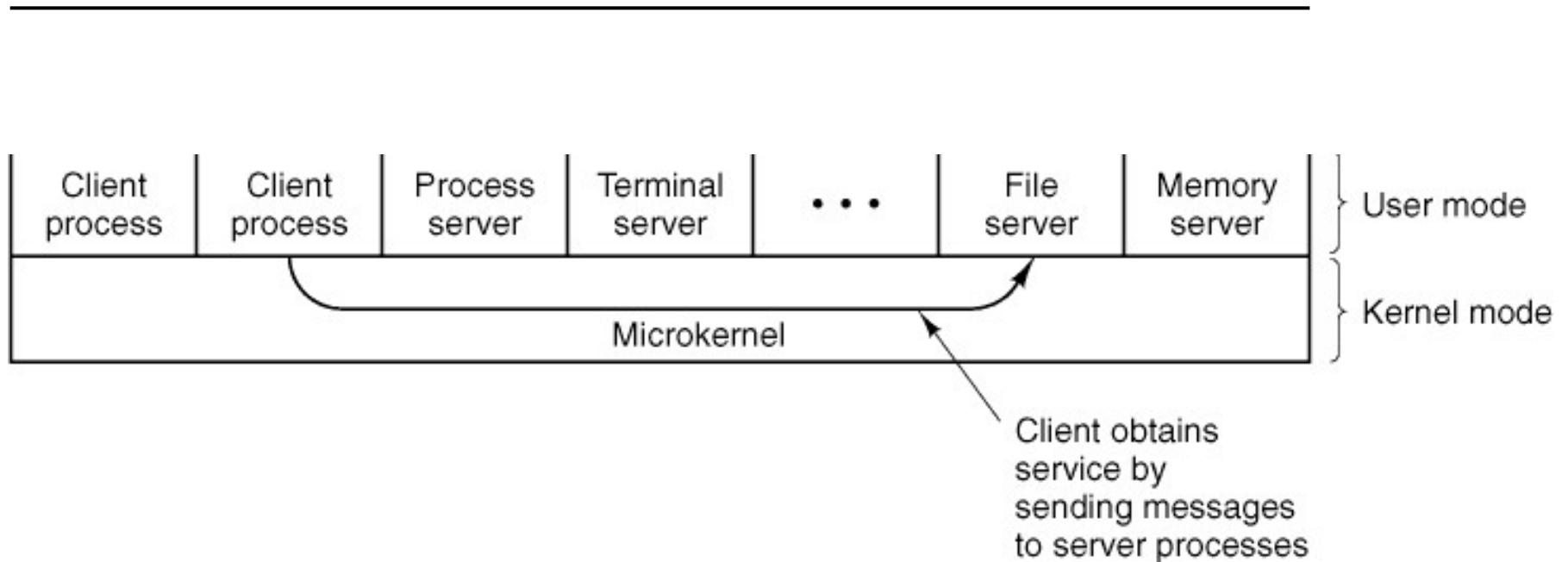
Layered System (2)

Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

Macchine Virtuali



Client-Server Model (1)



Client-Server Model (2)

