

Sistemi Operativi I & II

Matteo Vaccari <vaccari@pobox.com>

<http://matteo.vaccari.name/so/>

Testo:

Modern Operating Systems di Andrew Tanenbaum, Prentice-Hall

Ed. it: *Moderni Sistemi Operativi*, Jackson Libri

1

Argomenti del corso

- Introduzione
- Processi, thread e programmazione concorrente
- La gestione della memoria
- File system
- Input/output, periferiche e driver
- Sicurezza
- Linux & Unix

2

Questa lezione

Evoluzione dei SO

Funzioni di un SO

Struttura di un SO

3

Il Laboratorio di Sistemi Operativi

È parte integrante di questo corso

L'esame è in comune

Il voto è uno solo

Docente: prof. Alessandro Mazzetti

4

Modalità di esame

- a) Realizzazione di un elaborato: programma C
- b) Esame scritto su argomenti di S.O. I, II e Laboratorio
- c) Esame orale

Voto: media di a) e b), eventualmente modificata da c)

5

Il server Linux del laboratorio

È a disposizione un server Linux

Per chiedere la login compilate la form

Istruzioni sulla pagina web del corso

6

Perché studiare SO?

Per saper scrivere programmi applicativi

Per saper amministrare sistemi

Per saper scrivere driver di periferica

Per saper scrivere sistemi embedded

Per saper scrivere sistemi concorrenti o distribuiti

...per saper scrivere sistemi operativi!

7

Funzioni del Sistema Operativo

0. Estendere e astrarre l'hardware
per semplificare la programmazione
per rendere i programmi portabili
Esempio: un "file" è un astrazione

1. Gestire le risorse
suddividere stampanti, dischi, tempo di CPU fra più programmi

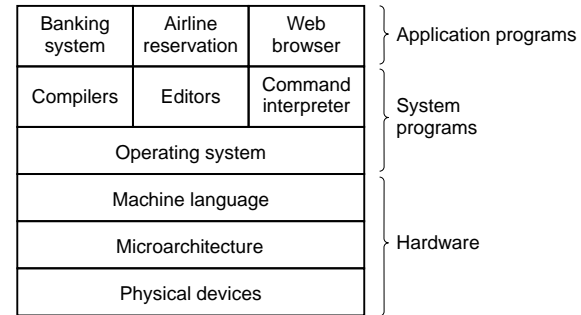
8

Funzioni del Sistema Operativo, II

in pratica:

0. Gestire i processi
1. Gestire la memoria
2. Gestire i file system
3. Gestire le periferiche
- (4. Gestire la rete)

9



A computer system consists of

- hardware
- system programs
- application programs

10

Un punto importante

Il sistema operativo esegue in modo *kernel*

Tutti gli altri programmi eseguono in modo *utente*

Modi di esecuzione della CPU:

- Kernel mode: tutte le istruzioni sono disponibili
- User mode: alcune istruzioni non sono disponibili
 - esempio: HALT

11

In cosa consiste il SO

Il *kernel* è il SO vero e proprio

È accompagnato da un insieme di *programmi di sistema*

Esempi: linker, compilatori, ...

init, mount, ifconfig, insmod, shutdown, ...

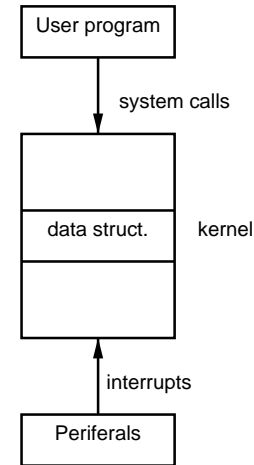
12

In cosa consiste il kernel

In prima approssimazione:

Il kernel è *un insieme di procedure*

13



14

Una chiamata di sistema (system call)

è una maniera di accedere ai servizi del SO

produce un salto da modo utente a modo kernel
in maniera controllata

per il programmatore è una chiamata di procedura

15

Esempio di chiamata di sistema: read(2)

Prototipo:

```
int read(int fd, void *buf, size_t count);
```

Esempio:

```
char buf[100];
int fd = open(pippo.txt, O_RDONLY);
int nr = read(fd, buf, 100);
```

Per la documentazione:

```
man 2 read
```

16

Questa *non* è una chiamata di sistema...

```
int fscanf(FILE *stream, const char *format, ...);
```

...è una chiamata *di libreria*

infatti si trova nella sezione 3 del manuale

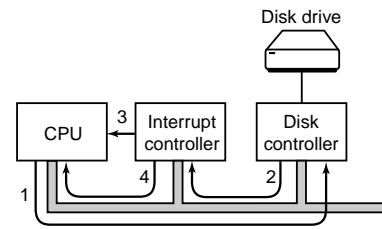
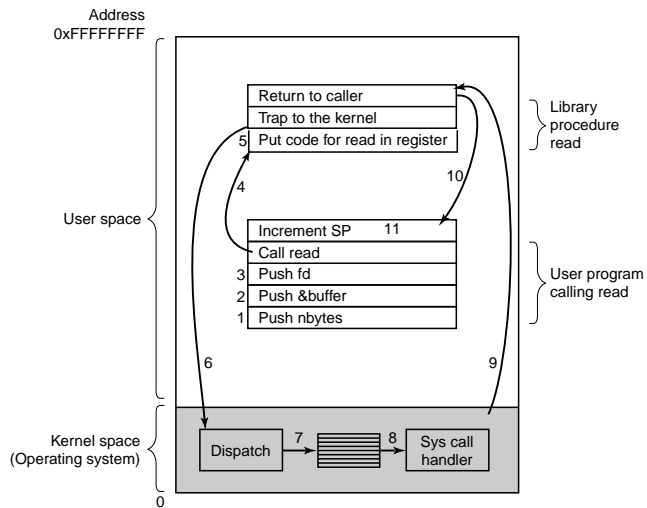
```
man 3 fscanf
```

Ma per funzionare, *deve* usare read(2)

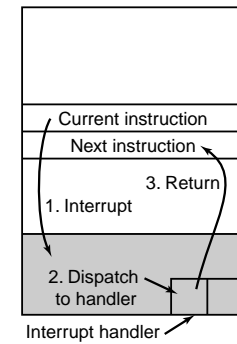
Neanche questa è una chiamata di sistema

```
size_t strlen(const char *s);
```

Di quali chiamate di sistema ha bisogno per funzionare?



(a)



(b)

Il manuale di Unix

Sezione 1: comandi per l'utente
"ls(1)", "man(1)", "who(1)"

Sezione 5: formati dei file
"fstab(5)", "printcap(5)"

Sezione 2: chiamate di sistema
"open(2)", "read(2)", "fork(2)"

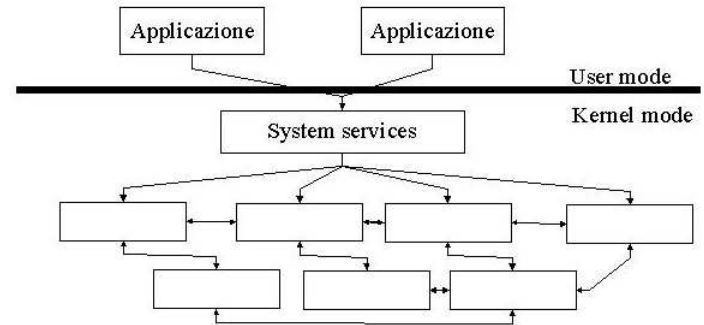
Sezione 6: giochi
"adventure(6)"

Sezione 3: chiamate di libreria
"fopen(3)", "scanf(3)", "strlen(3)"

Sezione 8: comandi per l'amministratore
"mount(8)"

21

Struttura monolitica



22

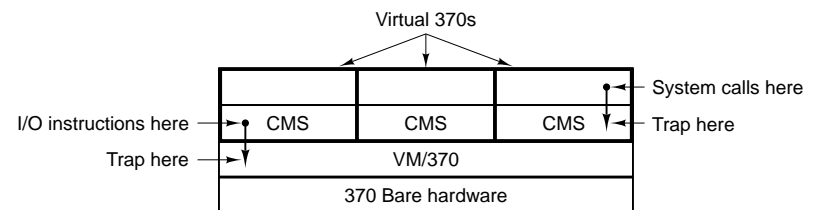
Struttura a strati (layered)

Esempio: il sistema THE di Dijkstra

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

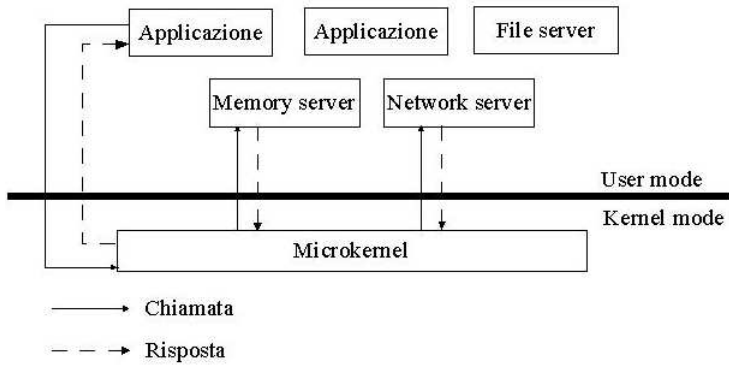
23

Struttura a macchina virtuale

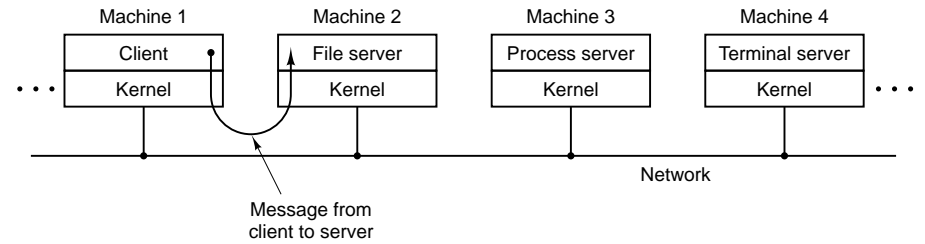


24

Struttura a microkernel (o client-server)



Struttura client-server in un sistema distribuito



Prima generazione (1945-1955)

Tecnologia: tubi elettronici

Input: plugboard, schede perforate

SO: librerie di subroutine

Si programma "la macchina nuda"

Seconda generazione (1955-1965)

Tecnologia: transistor

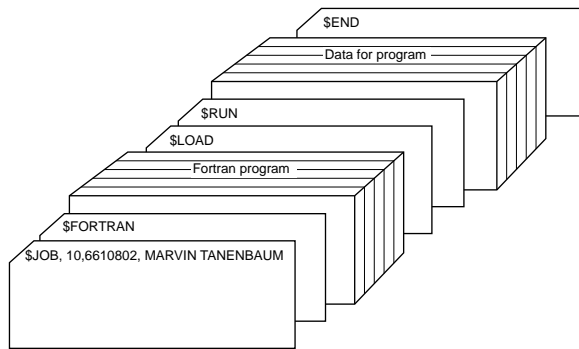
Input: schede perforate, nastri

SO: *monitor*

Elaborazione a *lotti* (batch)

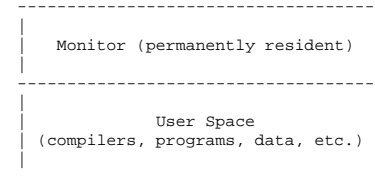
batch: una serie di *jobs*

Un tipico "job"



29

Il monitor



30

Il monitor, II

interpreta le istruzioni nel *batch control language*

carica i programmi e li esegue

mantiene informazioni di accounting

il programma utente ottiene *accesso totale alla macchina* fino a che non termina

31

Vantaggi dei sistemi a lotti (rispetto alla "macchina nuda")

- Il computer esegue molti compiti che erano dell'operatore
- Il computer esegue un nuovo job non appena è terminato il precedente

Problemi

- turn-around time can be large from user standpoint
- more difficult to debug program
- one batch job can affect pending jobs (read too many cards, etc)
- a job could corrupt the monitor, thus affecting pending jobs
- a job could enter an infinite loop

32

Soluzione a questi problemi: *protection hardware*

- la memoria del monitor diventa inaccessibile ai prog. utente
 - due *modi* di esecuzione: modo monitor e modo utente
 - I/O può essere fatto solo in modo monitor
-
- a ciascun processo si assegna un tempo massimo di esecuzione

33

Terza generazione (1965-1977)

Tecnologia: circuiti integrati

Spooling: Slow Peripheral Operation On-Line

Multiprogrammazione

Time-sharing

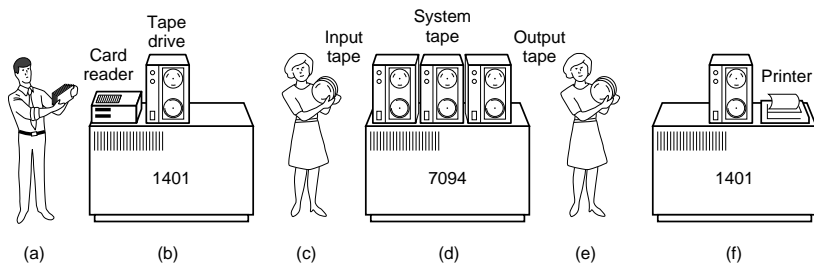
Minicomputer

OS/360

SO: moderno

34

Spooling Batch System



35

Multiprogrammazione

Il monitor diventa simile a un SO moderno

- fa I/O per i processi utente
- passa la CPU da un processo a un altro
- assicura la protezione fra i processi

36

Time-sharing

Ciascun *utente* ha l'illusione di avere tutta la macchina per sé

Corbató: MULTICS

Thompson, Ritchie: UNIX

VM/370

VAX/VMS

Windows NT

37

Quarta generazione (1977-presente)

Tecnologia: VLSI

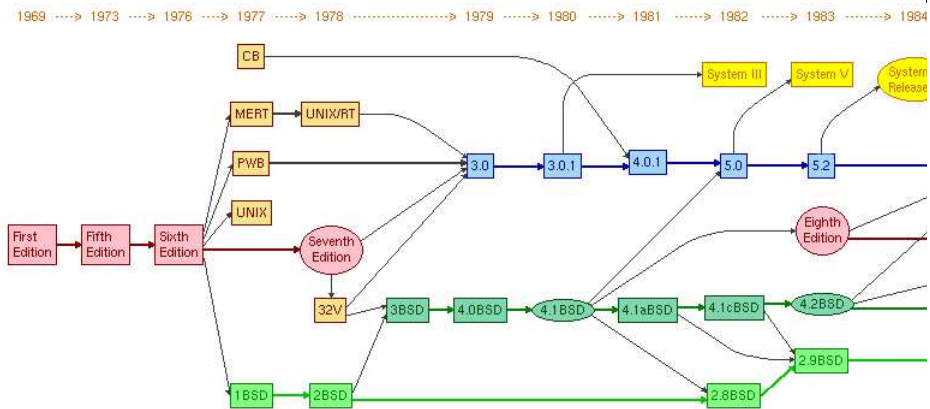
Personal computer

Interfacce utente grafiche

Sistemi distribuiti

SO: da "macchina nuda" a "moderno"

38



39

Ken Thompson e Dennis Ritchie



40