



DICo

Sistemi Operativi

Bruschi Monga

MINIX

System e kernel call

## Sistemi Operativi<sup>1</sup>

Mattia Monga

Dip. di Informatica e Comunicazione  
Università degli Studi di Milano, Italia  
[mattia.monga@unimi.it](mailto:mattia.monga@unimi.it)

a.a. 2009/10

<sup>1</sup> © 2010 M. Monga. Creative Commons Attribuzione-Condividi allo stesso modo 2.5 Italia License.  
<http://creativecommons.org/licenses/by-sa/2.5/it/>. Immagini tratte da [?] e da Wikipedia.

1



DICo

Sistemi Operativi

Bruschi Monga

MINIX

System e kernel call

## Lezione XVII: System e kernel call

314



DICo

Sistemi Operativi

Bruschi Monga

MINIX

System e kernel call

## MINIX assembly

- `man 9 as`
- `cc -S prova.c` produce l'assembly in `prova.s`
- `cc -o prova prova.s` produce l'eseguibile `prova`

315



DICo

Sistemi Operativi

Bruschi Monga

MINIX

System e kernel call

## Esercizi

- Scrivere un programma assembly che accetti un intero da tastiera e lo ristampi. È permesso fare uso delle chiamate della libreria standard del C `printf` e `scanf`
- Scrivere un programma assembly che contenga la definizione di una funzione con un parametro intero
- Scrivere un programma assembly che contenga la definizione di una funzione ricorsiva con un parametro intero

316

## Il boot monitor e il setup iniziale



Sistemi Operativi

Bruschi Monga

MINIX

System e kernel call

- Accedere al boot monitor
- Provare il comando help
- Cosa succede cambiando il parametro di boot memory=800... in memory=900...?
- Cambiare qualche campo di kinfo (p.es. release) modificandone l'inizializzazione in cstart

317

## System call



Sistemi Operativi

Bruschi Monga

MINIX

System e kernel call

In Minix le API del s.o. non sono vere syscall (i.e. girano in user space).

Ci sono tre tipologie di servizi del s.o.:

- 1 API POSIX, permesse anche ai processi utente
- 2 primitive di message passing, permesse solo ai componenti noti al s.o. (livelli 2 e 3)
  - send
  - receive
  - notify
  - reply
- 3 kernel call permesse solo ai componenti noti al s.o. (livelli 2 e 3)

318

## MINIX

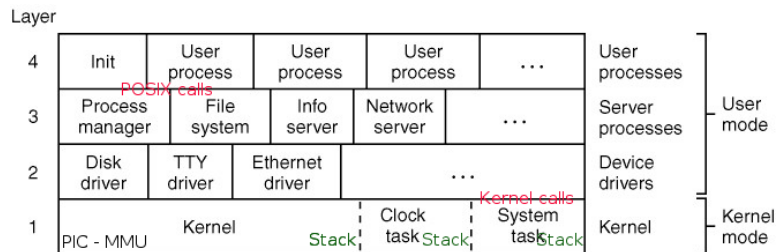


Sistemi Operativi

Bruschi Monga

MINIX

System e kernel call



319

## API POSIX



Sistemi Operativi

Bruschi Monga

MINIX

System e kernel call

Una chiamata ad una syscall POSIX scatena uno scambio di messaggio che provoca a sua volta una (o più) kernel call, le uniche chiamate in grado di manipolare i dati del kernel

- 1 Chiamata a fork()
- 2 Messaggio opportuno a PM ("ho bisogno della syscall fork")
- 3 Messaggio SYS\_FORK al System Task
- 4 Esecuzione di do\_fork()

320

# Messaggi



Sistemi Operativi  
Bruschi Monga

MINIX  
System e kernel call

```

1 /* minix/ipc.h */
2
3 typedef struct {
4     int m_source; /* who sent the message */
5     int m_type; /* what kind of message is it */
6     union {
7         mess_1 m_m1;
8         mess_2 m_m2;
9         mess_3 m_m3;
10        mess_4 m_m4;
11        mess_5 m_m5;
12        mess_7 m_m7;
13        mess_8 m_m8;
14    } m_u;
15 } message;

1 /* minix/ipc.h */
2 typedef struct {int m1i1, m1i2, m1i3; char *m1p1, *m1p2, *m1p3;} mess_1;
3 /* ... */
    
```

# Esperimento



Sistemi Operativi  
Bruschi Monga

MINIX  
System e kernel call

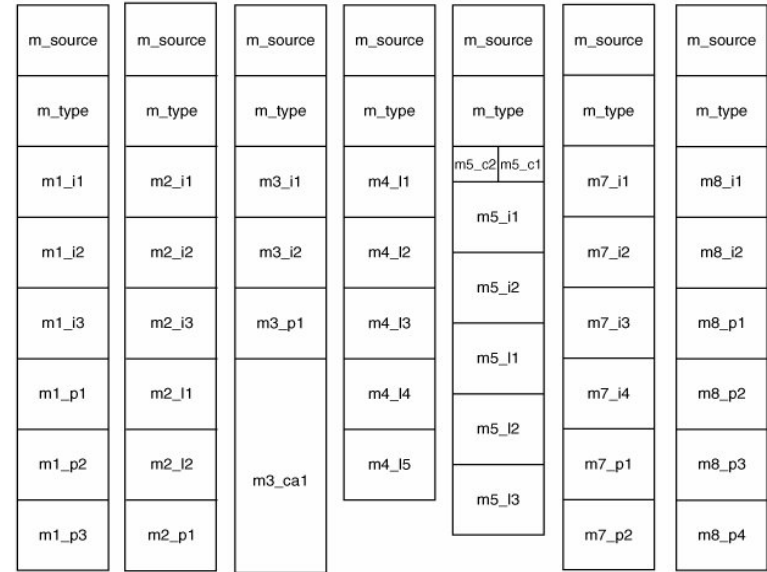
Scrivere un programma che chiama direttamente una primitiva di message passing, per esempio send. Il prototipo si trova in ipc.h, come destinazione usare ANY

# Messaggi



Sistemi Operativi  
Bruschi Monga

MINIX  
System e kernel call



# Esercizi



Sistemi Operativi  
Bruschi Monga

MINIX  
System e kernel call

- 1 Stampare un messaggio all'esecuzione di ogni exit
- 2 Stampare un messaggio all'esecuzione di ogni exit con il numero di exit eseguite fino a quel momento
- 3 Stampare un messaggio all'esecuzione di ogni exit con il numero di API del s.o. eseguite fino a quel momento
- 4 Stampare il nome del programma caricato in memoria quando viene eseguita una exec