Sistemi Operativi (Laboratorio)

Lorenzo Martignoni

Dipartimento di Informatica e Comunicazione
Università degli Studi di Milano, Italia
lorenzo@security.dico.unimi.it

a.a. 2008/09

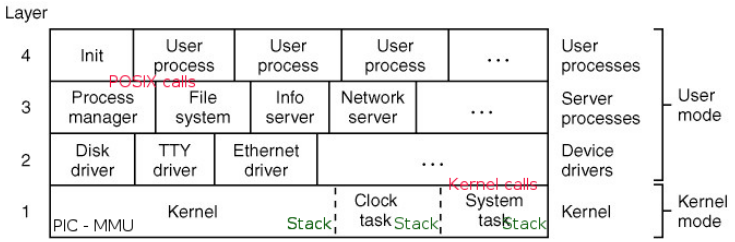
Lezione VII: System e kernel call

System 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)

MINIX



API POSIX

Una chiamata ad una syscall POSIX scatena uno scambio di messaggio che provoca a sua volta una (o piú) 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()`

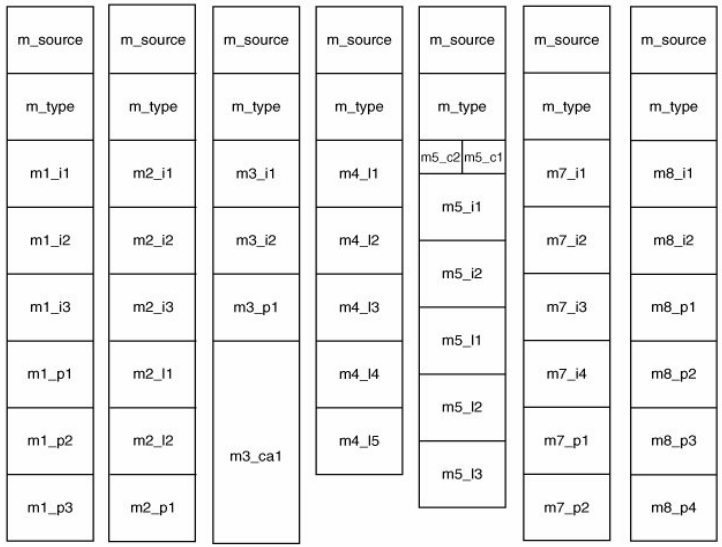
Messaggi

```
/* minix/ipc.h */

typedef struct {
  int m_source; /* who sent the message */
  int m_type; /* what kind of message is it */
  union {
    mess_1 m_m1;
    mess_2 m_m2;
    mess_3 m_m3;
    mess_4 m_m4;
    mess_5 m_m5;
    mess_7 m_m7;
    mess_8 m_m8;
  } m_u;
} message;

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


Messaggi



Esperimento

Scrivere un programma che chiama direttamente una system call, per esempio `kill`. Il prototipo si trova in `ipc.h`.

Potete trovare uno spunto in `/usr/src/lib/posix`.

Esperimento

Convertire il programma precedente per chiamare direttamente una primitiva di message passing. Il prototipo si trova in `ipc.h`.

Potete trovare uno spunto in `/usr/src/lib/other/syscall.c`.

Esercizi

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`

Come aggiungere una system call `foo()`

Supponendo che il servizio sia gestito gestito da PM

1. Aggiornare `/usr/src/include/minix/callnr.h`
2. Aggiornare tutti i `call_vec` (`no_sys` in FS e `do_foo` in PM)
3. Aggiungere `do_foo()` (p.es. in `pm/misc.c`)
4. Creare la libreria wrapper e aggiornare `unistd.h`

Esercizi

- Aggiungere una syscall che stampa "ciao"
- Aggiungere una syscall che stampa il numero di processi nella coda ready

Copyright