Sistemi Operativi (Laboratorio)

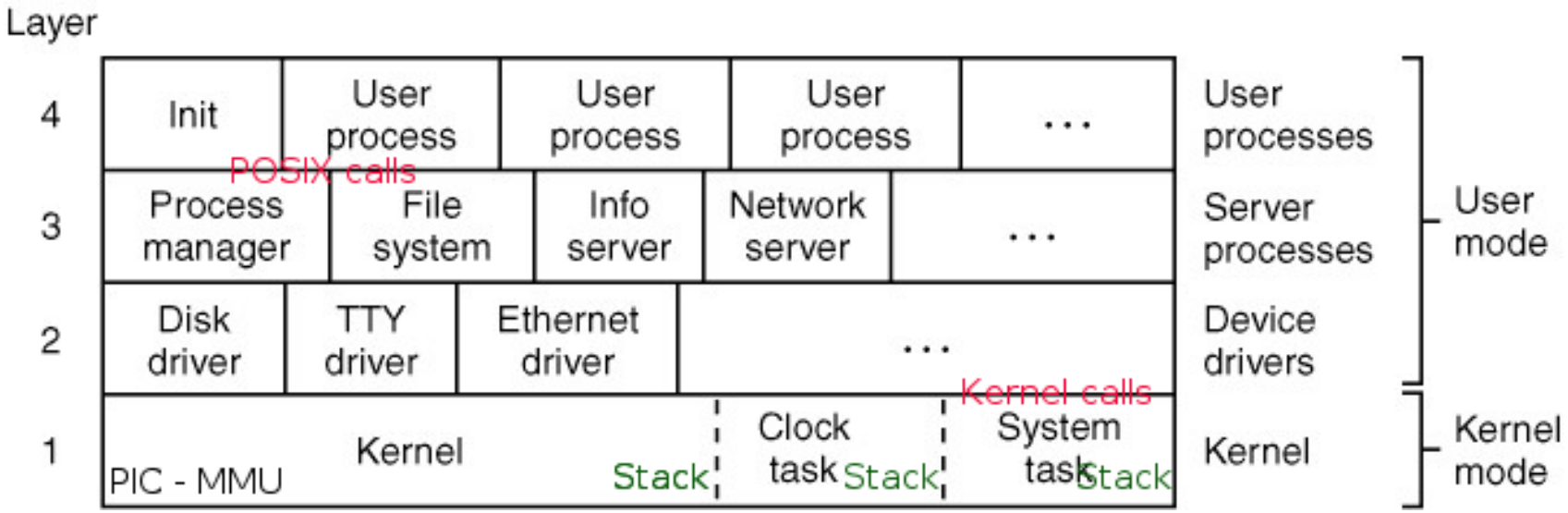
Lorenzo Martignoni

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

a.a. 2008/09

Lezione XI: System call

MINIX


Layer
4
3
2
1
Init
User process
User process
User process
...
POSIX calls
Process manager
File system
Info server
Network server
...
Disk driver
TTY driver
Ethernet driver
...
Kernel calls
PIC - MMU
Kernel
Stack
Clock task
Stack
System task
Stack
User processes
Server processes
Device drivers
Kernel
User mode
Kernel mode

Un semplice semaforo

Primitive da implementare:

1. sem_init
2. sem_status
3. sem_up
4. sem_down

Aggiungiamo queste primitive (system call) ad un server esistente (FS).

Un semplice semaforo

```
int do_down(message *m_ptr) {
        /* Resource available. Decrement semaphore and reply. */
        if (s > 0) {
            s = s  1; /* take a resource */
            return(OK); /* let the caller continue */
        }
        /* Resource taken. Enqueue and block the caller. */
        enqueue(m_ptr->m_source); /* add process to queue */
        return(EDONTREPLY); /* do not reply in order to block
the caller */
}

int do_up(message *m_ptr) {
    message m; /* place to construct reply message */
        /* Add resource, and return OK to let caller continue. */
        s = s + 1; /* add a resource */
        /* Check if there are processes blocked on the semaphore. */
        if (queue_size() > 0) { /* are any processes blocked? */
            m.m_type = OK;
            m.m_source = dequeue(); /* remove process from queue */
            s = s  1; /* process takes a resource */
            send(m.m_source, m); /* reply to unblock the process */
        }
        return(OK); /* let the caller continue */
    }
```

Semaphore Server

Creare un nuovo server che implementa le primitive per la gestione dei semafori

Semaphore Server

Aggiungere un server SS che gestisca un semaforo (s) secondo lo schema indicato.

- Partire da un server esistente (p.es. DS o IS)
- Aggiornare con i messaggi possibili `include/minix/com.h`
- Aggiungere il server alla boot image (è piú semplice che caricarlo poi)
- Aggiornare i makefile

Semaphore Server

```
void semaphore_server( ) {
    message m;
    int result;
    /* Initialize the semaphore server. */
    initialize( );
    /* Main loop of server. Get work and process it. */
    while(TRUE) {
        /* Block and wait until a request message arrives. */
        get_work(&m);
        /* Caller is now blocked. Dispatch based on message type. */
        switch(m.m_type) {
            case UP: result = do_up(&m); break;
            case DOWN: result = do_down(&m); break;
            default: result = EINVAL;
        }
        /* Send the reply, unless the caller must be blocked. */
        if (result != EDONTREPLY) {
            m.m_type = result;
            send(m.m_source, &m);
        }
    }
}
```