Sistemi Operativi[1]

Mattia Monga

Dip. di Informatica
Università degli Studi di Milano, Italia
mattia.monga@unimi.it

a.a. 2011/12

Lezione XXVIII: Esercizi kernel

Un server "semafori"


```
1  void semaphore_server( ) {
2      message m;
3      int result;
4      /* Initialize the semaphore server. */
5      initialize( );
6      /* Main loop of server. Get work and process it. */
7      while(TRUE) {
8          /* Block and wait until a request message arrives. */
9          get_work(&m);
10         /* Caller is now blocked. Dispatch based on message type. */
11         switch(m.m_type) {
12             case UP: result = do_up(&m); break;
13             case DOWN: result = do_down(&m); break;
14             default: result = EINVAL;
15         }
16         /* Send the reply, unless the caller must be blocked. */
17         if (result != EDONTREPLY) {
18             m.m_type = result;
19             send(m.m_source, &m);
20         }
21     }
22 }
```

Un server "semafori"


```
1  int do_down(message *m_ptr) {
2          /* Resource available. Decrement semaphore and reply. */
3          if (s > 0) {
4                  s = s - 1; /* take a resource */
5                  return(OK); /* let the caller continue */
6          }
7          /* Resource taken. Enqueue and block the caller. */
8          enqueue(m_ptr->m_source); /* add process to queue */
9          return(EDONTREPLY); /* do not reply in order to block
10 the caller */
11 }
12 int do_up(message *m_ptr) {
13     message m; /* place to construct reply message */
14         /* Add resource, and return OK to let caller continue. */
15         s = s + 1; /* add a resource */
16         /* Check if there are processes blocked on the semaphore. */
17         if (queue_size() > 0) { /* are any processes blocked? */
18                 m.m_type = OK;
19                 m.m_source = dequeue(); /* remove process from queue */
20                 s = s - 1; /* process takes a resource */
21                 send(m.m_source, m); /* reply to unblock the process */
22         }
23         return(OK); /* let the caller continue */
24     }
```

Esercizio

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

Esercizi server SS

- Aggiungere un comando di inizializzazione del semaforo con un valore dato
- Modificare il server in modo che sia possibile creare piú semafori, ciascuno identificato da un numero intero

Esercizi server SS

- Utilizzando il server semafori implementare la seguente sincronizzazione:

  Un processo `docente` risponde alle domande di alcuni processi `studente`, con i seguenti vincoli:

  1. in ogni momento un solo processo (docente o studente) emette messaggi (domande e risposte ottenute con printf sullo stdout)
  2. ogni domanda degli studenti ottiene una specifica risposta
  3. l'elaborazione della risposta da parte del docente richiede 2 secondi
  4. ciascuno studente si astiene dal fare una nuova domanda finché il docente non ha risposta alla precedente
  5. Ogni studente fa un'unica domanda

  Esempio di output

```
% student: ask question
% prof: answer question
% student: leave room
% student: ask question
% prof: answer question
% student: leave room
% student: ask question
% prof: answer question
% student: leave room
```

Strutture globali

1. Aggiungere la struttura dati (p.es. in `kernel/proc.h`
2. Dichiarare, definire ed inizializzare la struttura dati globale
3. Definire l'aggiornamento
4. Definirne l'uso. Per esempio:
   - Alla pressione del tasto F8 il contenuto della struttura dati viene stampato sullo schermo