YAEOS

Yet Another Educational Operating System
Specifiche dell'esercitazione di Laboratorio
Sistemi Operativi
A.A. 2017-18

Phase1

Renzo Davoli
(alcuni lucidi sono tratti da un documento di Marco di Felice)

YAEOS

- Da svolgersi su μARM (ultima versione da GIT)
- E' l'ultimo di una lunga serie di progetti didattici (hoca, tina, icaros, mikaboo, ...)

YAEOS

- Livello 6: Shell interattiva
- Livello 5: File--system
- Livello 4: Livello di supporto
- Livello 3: Kernel del S.O.
- Livello 2: Gestione delle Code
- Livello 1: Servizi offerti dalla ROM
- Livello 0: Hardware di uMPS2

Il livello 0 e 1 sono già disponibili

Il livello 2 è la fase 1 del progetto

Il livello 3 è la fase 2 del progetto

Phase1 = Livello 2

- Tutte le funzioni devono essere implementate con **scansioni ricorsive** (è vietata l'iterazione)

Livello 2: PCB

```
Typedef struct pcb_t  {
  struct pcb_t *p_next;
  struct pcb_t *p_parent;
  struct pcb_t *p_first_child;
  struct pcb_t *p_sib;
  state_t p_s;
  int priority;
  int *p_semKey;
} pcb_t;
```

Livello 2: PCB

- Il gestore delle code implementa quattro funzionalità relative ai PCB:
  - Allocazione e deallocazione dei PCB.
  - Gestione delle code dei PCB.
  - Gestione dell'abero dei PCB.
  - Gestone di una Active Semaphore Hash Table (ASHT), che gestisce i processi bloccati sui semafori.
- ASSUNZIONE: non ci sono più di MAXPROC processi concorrenti in YAEOS.

Allocazione dei PCB

- pcbFree: lista dei PCB che sono liberi o inutilizzati.
  - `pcb_t *pcbfree_h`: testa della lista pcbFree.
  - `pcb_t pcbFree_table[MAXPROC]`: array di PCB con dimensione MAXPROC.

Funzioni da implementare

- `void initPcbs()`
  - DESCRIZIONE: Inizializza la pcbFree in modo da contenere tutti gli elementi della pcbFree_table. Questo metodo deve essere chiamato una volta sola in fase di inizializzazione della struttura dati.

Funzioni da implementare

- `void freePcb(pcb_t *p)`
  - DESCRIZIONE: Inserisce il PCB puntato da p nella lista dei PCB liberi (pcbFree)
- `pcb_t *allocPcb()`
  - DESCRIZIONE: Restituisce NULL se la pcbFree è vuota. Altrimenti rimuove un elemento dalla pcbFree, inizializza tutti i campi (NULL/0) e restituisce l'elemento rimosso.

Funzioni da implementare

- `void insertProcQ(pcb_t **head, pcb *p)`

  DESCRIZIONE: inserisce l'elemento puntato da p nella coda dei processi puntata da head. L'inserimento deve avvenire tenendo conto della priorità di ciascun pcb (campo p-->priority). La coda dei processi deve essere ordinata in base alla priorità dei PCB, in ordine decrescente (i.e. l'elemento di testa è l'elemento con la priorità più alta).

  `pcb_t headProcQ(pcb_t *head)`

  DESCRIZIONE: Restituisce l'elemento di testa della coda dei processi da head, SENZA RIMUOVERLO. Ritorna NULL se la coda non ha elementi.

Funzioni da implementare

- `pcb_t* removeProcQ(pcb_t **head)`

  DESCRIZIONE: rimuove il primo elemento dalla coda dei processi puntata da head. Ritorna NULL se la coda è vuota. Altrimenti ritorna il puntatore all'elemento rimosso dalla lista.
- `pcb_t* outProcQ(pcb_t **head, pcb_t *p)`

  DESCRIZIONE: Rimuove il PCB puntato da p dalla coda dei processi puntata da head. Se p non è presente nella coda, restituisce NULL. (NOTA: p può trovarsi in una posizione arbitraria all'interno della coda).
- ```
  void forallProcQ(pcb_t *head,
   void *fun(pcb_t *pcb, void *),
   void *arg)
  ```

  richiama la funzione fun per ogni elemento della lista puntata da head.
  ```

Alberi di PCB

- Oltre a poter appartenere appartenere ad una coda di processi, i PCB sono organizzati in alberi di processi.
- Ogni genitore contiene un puntatore (p_child) alla testa della lista dei figli.
- Ogni figlio ha un puntatore al genitore (p_parent) e uno che punta al successivo fratello (p_sibling).

Funzioni da implementare

- `void insertChild(pcb_t *parent, pcb_t *p)`

  DESCRIZIONE: Inserisce il PCB puntato da p come figlio del PCB puntato da parent.
- `pcb_t *removeChild(pcb_t *p)`

  DESCRIZIONE. Rimuove il primo figlio del PCB puntato da p. Se p non ha figli, restituisce NULL.
- `pcb_t *outChild(pcb_t* p)`

  DESCRIZIONE: Rimuove il PCB puntato da p dalla lista dei figli del padre. Se il PCB puntato da p non ha un padre, restituisce NULL. Altrimenti restituisce l'elemento rimosso (cioé p). A differenza della removeChild, p può trovarsi in una posizione arbitraria (ossia non è necessariamente il primo figlio del padre).

Semafori

In YAEOS, l'accesso alle risorse condivise avviene attraverso l'utilizzo di semafori.

- Ad ogni semaforo è associato un descrittore (SEMD) con la struttura seguente:

```
typedef struct semd_t {
    struct semd_t *s_next;
    int *s_key;
    struct pcb_t *s_procQ;
} semd_t;
```

s_key è l'indirizzo della variabile intera che contiene il valore del semaforo. L'indirizzo di s_key serve come identificatore del semaforo.

Active Semaphore Hash Table (ASHT)

- `semd_t semd_table[MAXSEMD]`: array di SEMD con dimensione massima MAXSEMD.
- `semd_t *semdFree_h`: Testa della lista dei SEMD liberi o inutilizzati. I SEMD attivi sono gestiti tramite una Hash Table di dimensione ASHDSIZE
- `semd_t *semdhash[ASHDSIZE]`: hash table, (ogni elemento punta alla lista di collisione per il valore di hash corrispondente all'indice)

Funzioni da implementare

- `int insertBlocked(int *key,pcb_t *p)`

  DESCRIZIONE: Viene inserito il PCB puntato da p nella coda dei processi bloccati associata al semaforo con chiave key. Se il semaforo corrispondente non è presente nella ASHT, alloca un nuovo SEMD dalla lista di quelli liberi e lo inserisce nella ASHT, settando i campi in maniera opportuna. Se non è possibile allocare un nuovo SEMD perché la lista di quelli liberi e' vuota, restituisce -1. In tutti gli altri casi, restituisce 0.
- `pcb_t *headBlocked(int *key)`

  DESCRIZIONE: restituisce il puntatore al pcb del primo processo bloccato sul semaforo, senza deaccordarlo. Se il semaforo non esiste restituisce NULL.

Funzioni da implementare

- `pcb_t* removeBlocked(int  *key)`

  DESCRIZIONE: Ritorna il primo PCB dalla coda dei processi bloccati (s_ProcQ) associata al SEMD della ASHT con chiave key. Se tale descrittore non esiste nella ASHT, restituisce NULL. Altrimenti, restituisce l'elemento rimosso. Se la coda dei processi bloccati per il semaforo diventa vuota, rimuove il descrittore corrispondente dalla ASHT e lo inserisce nella coda dei descrittori liberi (semdFree).
- ```
  void forallBlocked(int *key,
     void (*fun)(pcb_t *pcb, void *),
     void *arg)
  ```

  richiama la funzione fun per ogni processo bloccato sul semaforo identificato da key.
  ```

Funzioni da implementare

- `outChildBlocked(pcb_t *p)`

  DESCRIZIONE: Rimuove il PCB puntato da p dalla coda del semaforo su cui è bloccato. (La hash table deve essere aggiornata in modo coerente).
- `void initASL()`

  DESCRIZIONE: Inizializza la lista dei semdFree in modo da contenere tutti gli elementi della semdTable. Questo metodo viene invocato una volta sola durante l'inizializzazione della struttura dati.

Consegna

- La deadline di consegna è fissata per il giorno: Domenica 18 Febbraio 2018, ore 23.59
- CONSEGNARE IL PROPRIO PROGETTO (un unico file .tar.gz) NELLA DIRECTORY DI CONSEGNA ASSOCIATA AL PROPRIO GRUPPO:
- `/home/students/LABSO/2018/submit_phase1/lso2018az...`
- CONSEGNARE ENTRO LA DEADLINE FISSATA.
- VERIFICARE CHE L'ARCHIVIO .TAR.GZ SIA COMPLETO

Consegna

- Cosa consegnare:
  - Sorgenti del progetto (TUTTI)
  - Makefile per la compilazione (eventuali file di AutoMake/Cmake)
  - README con istruzioni di compilazione
  - Documentazione (scelte progettuali)
  - File AUTHORS

- Occorre inserire commenti nel codice per favorire la leggibilità e la correzione …
- PROGETTI non COMMENTATI NON SARANNO VALUTATI.
- PROGETTI che contengono troppi commenti inutili verranno valutati negativamente
- VERRÀ valutato il grado di professionalità nella gestione del codice