SOS

Strange Operating System
Specifiche dell'esercitazione di Laboratorio
Sistemi Operativi
A.A. 2012-13

Phase1

Renzo Davoli
(sulla base dei lucidi di Marco Di Felice)

SOS

- SOS: Evoluzione di Kaya O.S., a sua volta evoluzione di una lunga lista di S.O. propostp a scopo didattico (HOCA, TINA, ICARO, etc).
- Caratteristiche del progetto SOS:
- Da svolgersi esclusivamente su architettura uMPS2
- Disegnato per sistema multiprocessore
-

SOS

- 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 gia' disponibili

Il livello 2 e' la fase 1 del progetto

Il livello 3 e' la fase 2 del progetto

Phase1 = Livello 2

- Tutte le funzioni devono essere implementate con scansioni ricorsive (no iterazione)

Livello 2:

- (PCB) livello 2.

```
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 dei PCB

- Il gestore delle code implementa 4 funzionalita' relative ai PCB:
  - Allocazione e Deallocazione dei PCB.
  - Gestione delle Code dei PCB.
  - Gestione della lista dei PCB.
  - Gestone di una Active Semaphore List (ASL), che gestisce la coda dei processi bloccati su un semaforo.
- ASSUNZIONE: non ci sono piu' di MAXPROC processi concorrenti in PKaya.

Allocazione dei PCB

- pcbFree: lista dei PCB che sono liberi o inutilizzati.
  - pcbfree_h: testa della lista pcbFree.
  - pcb_table[MAXPROC]: array di PCB con dimensione massima di MAX_PROC.

Funzioni da implementare

- void initPcbs()
  - DESCRIZIONE: Inizializza la pcbFree in modo da contenere tutti gli elementi della pcb_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 e' 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_t* p)

DESCRIZIONE: inserisce l'elemento puntato da p nella coda dei processi puntata da head. L'inserimento deve avvenire tenendo conto della priorita' di ciascun pcb (campo p->priority). La coda dei processi deve essere ordinata in base alla priorita' dei PCB, in ordine decrescente (i.e. l'elemento di testa e' l'elemento con la priorita' piu' 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 e' 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 e' presente nella coda, restituisce NULL. (NOTA: p puo' trovarsi in una posizione arbitraria della coda).

Void forallProcQ(struct pcb_t *head, void *fun(struct pcb_t *pcb, void *), void *arg)

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

Alberi di PCB

- In aggiunta alla possibilita' di partecipare ad una coda di processo, i PCB sono essere organizzati in alberi di processi .
- Ogni genitore contiene un puntatore (p_child) che punta alla lista dei figli.
- Ogni figlio ha un puntatore al padre (p_parent) e uno che punta al successivo fratello.

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 (cioe' p). A differenza della removeChild, p puo' trovarsi in una posizione arbitraria (ossia non e' necessariamente il primo figlio del padre).

Semafori

In SOS, l’accesso alle risorse condivise avviene attraverso l’utilizzo di semafori.

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

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

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

Active Semaphore List

- semd_table[MAXPROC]: array di SEMD con dimensione massima di MAXPROC.
- semdFree_h: Testa della lista dei SEMD liberi o inutilizzati.
- E’ necessario gestire la lista dei SEMD attivi (Active Semaphore List – ASL)
- semd_h: puntatore alla testa dei semafori attivi

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 SEMD con chiave key. Se il semaforo corrispondente non e' presente nella ASL, alloca un nuovo SEMD dalla lista di quelli liberi (semdFree) e lo inserisce nella ASL, settando I campi in maniera opportuna. Se non e' possibile allocare un nuovo SEMD perche' la lista di quelli liberi e' vuota, restituisce TRUE. In tutti gli altri casi, restituisce FALSE.

Funzioni da implementare

pcb_t* removeBlocked(int *key)

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

Void forallBlocked(int *key, void *fun(struct 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 e' bloccato

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 strutuura dati.

Consegna

- La deadline di consegna e’ fissata per il giorno: Domenica 17 Febbraio 2013, ore 23.59
- CONSEGNARE IL PROPRIO PROGETTO (un unico file .tar.gz) NELLA CARTELLA DI CONSEGNA ASSOCIATA AL PROPRIO GRUPPO:

  /home/students/LABSO/2013/**submit_phase1**/lso2013az...
- CONSEGNARE ENTRO LA DEADLINE FISSATA.
- VERIFICARE CHE L’ARCHIVIO .TAR.GZ NON SIA ROVINATO.

Consegna

- Cosa consegnare:
  - Sorgenti del progetto (TUTTI)
  - Makefile per la compilazione
  - Documentazione (scelte progettuali)
  - File AUTHORS

- Inserire commenti nel codice per favorire la leggibilita' e la correzione …
- PROGETTI non COMMENTATI NON SARANNO VALUTATI.
-
-