STEFANO DE SANTIS, CRISTIANO GUIDOTTI, IACOPO PORCEDDA, JACOPO RIMEDIOTTI
Documentazione fase 1 progetto BiKaya del corso Sistemi Operativi
Anno Accademico 2019-2020

Documentazione e scelte progettuali

Scelte progettuali

La fase 1 del progetto BiKaya è stata divisa nello sviluppo in linguaggio C di due moduli: Process Control Block e Active Semaphore List.
Entrambi compatibili per entrambe le architetture uARM e uMPS.

Entrambe le strutture dati sono allocate staticamente su un array di dimensione costante MAXPROC ed è possibile accedere ai nodi liberi delle strutture dati, tramite le variabili sentinelle pcbFree_h e semdFree_h, associate alle liste dei PCB e SEMD liberi.

Entrambe saranno significative e accessibili solo dopo che sono state chiamate rispettivamente le funzioni initPcbs e initASL.

Process Control Block (PCB)

Il PCB è una struttura dati associabile ad una coda di PCB contenenti alberi di PCB.

La coda PCB è implementata come coda basata su priorità, dove ad ogni elemento PCB è associato un albero dei PCB figli, implementato però come una normale coda dove ogni nodo è un PCB fratello.

In entrambe le situazioni, le code sono implementate con liste doppie circolari.

Active Semaphore List (ASL)

La ASL è una struttura dati associabile ad una lista di SEMD (Semaphore Descriptor) contenenti code di PCB.

La lista ASL è implementata con una lista doppia circolare, i cui elementi contengono anche la sentinella della coda PCB associata.

STEFANO DE SANTIS, CRISTIANO GUIDOTTI, IACOPO PORCEDDA, JACOPO RIMEDIOTTI
Documentazione fase 1 progetto BiKaya del corso Sistemi Operativi
Anno Accademico 2019-2020

Documentazione

pcb.h

- void initPcbs()
  - 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.
- pcb_t* allocPcb()
  - Restituisce NULL se la pcbFree è vuota, altrimenti rimuove un elemento dalla pcbFree. inizializza tutti i campi (NULL/0) e restituisce l'elemento rimosso.
- void freePcb(pcb_t *p)
  - Inserisce il PCB puntato da p nella lista dei PCB liberi (pcbFree)
- void mkEmptyProcQ(struct list_head *head)
  - Inizializza la lista dei PCB, inizializzando l'elemento sentinella.
- int emptyProcQ(struct list_head *head)
  - Restituisce TRUE se la lista puntata da head è vuota, FALSE altrimenti.
- void insertProcQ(struct list_head *head, pcb_t *p)
  - Inserisce l'elemento puntato da p nella coda dei processi puntata da head.
    L'inserimento avvenire tenendo conto della priorità di ciascun pcb (campo p->priority).
    La coda dei processi è ordinata in modo decrescente (i.e. l'elemento di testa è l'elemento con la priorità più alta).
- pcb_t *removeProcQ(struct list_head *head)
  - 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 *headProcQ(struct list_head *head)
  - Restituisce l'elemento di testa della coda dei processi da head, SENZA RIMUOVERLO.
    Ritorna NULL se la coda non ha elementi.
- pcb_t *outProcQ(struct list_head *head, pcb_t *p)
  - 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)
- int emptyChild(pcb_t *this)
  - restituisce TRUE se il PCB puntato da p non ha figli, restituisce FALSE altrimenti.
- void insertChild(pcb_t *prnt, pcb_t *p)
  - Inserisce il PCB puntato da p come figlio del PCB puntato da prnt.
- pcb_t *removeChild(pcb_t *p)
  - Rimuove il primo figlio del PCB puntato da p.
    Se p non ha figli, restituisce NULL
- pcb_t *outChild(pcb_t *p)
  - 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 (cioè p). A differenza della removeChild, p può trovarsi in una posizione arbitraria (ossia non è necessariamente il primo figlio del padre)

STEFANO DE SANTIS, CRISTIANO GUIDOTTI, IACOPO PORCEDDA, JACOPO RIMEDIOTTI
Documentazione fase 1 progetto BiKaya del corso Sistemi Operativi
Anno Accademico 2019-2020

asl.h

- void initASL()
  - Inizializza ASL
- semd_t* getSemd(int *key)
  - Restituisce il puntatore al SEMD nella ASL la cui chiave è pari a key. Se non esiste un elemento nella ASL con chiave eguale a key, viene restituito NULL.
- int insertBlocked(int *key,pcb_t* p)
  - Viene inserito il PCB puntato da p nella coda dei processi bloccati associata al SEMD con chiave key.
    Se il semaforo corrispondente non è presente nella ASL, alloca un nuovo SEMD dalla lista di quelli liberi (semdFree) e lo inserisce nella ASL, settando I campi in maniera opportuna (i.e. key e s_procQ).
    Se non è possibile allocare un nuovo SEMD perché la lista di quelli liberi è vuota, restituisce TRUE.
    In tutti gli altri casi, restituisce FALSE.
- pcb_t* removeBlocked(int *key)
  - 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).
- pcb_t* outBlocked(pcb_t *p)
  - Rimuove il PCB puntato da p dalla coda del semaforo su cui è bloccato (indicato da p->p_semKey).
    Se il PCB non compare in tale coda, allora restituisce NULL (condizione di errore).
    Altrimenti, restituisce p.
    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).
- pcb_t* headBlocked(int *key)
  - Restituisce (senza rimuovere) il puntatore al PCB che si trova in testa alla coda dei processi associata al SEMD con chiave key.
    Ritorna NULL se il SEMD non compare nella ASL oppure se compare ma la sua coda dei processi è vuota.
- void outChildBlocked(pcb_t *p)
  - Rimuove il PCB puntato da p dalla coda del semaforo su cui è bloccato (indicato da p->p_semKey).
    Inoltre, elimina tutti i processi dell'albero radicato in p (ossia tutti i processi che hanno come avo p) dalle eventuali code dei semafori su cui sono bloccati.