Progetto di Sistemi Operativi

PandOS Livello 2

Ahmed · Niouer
Matricola: 0001027288

Manuel · Bacci
Matricola: 0001040090

Mattia · Malservigi
Matricola: 0001029225

Shanshan · Feng
Matricola: 0000890555

Anno accademico
2022 – 2023

Corso di Sistemi Operativi
Renzo Davoli
Università di Bologna

1 INTRODUZIONE

`PandOS+` è un particolare sistema operativo proposto a scopo didattico. Il S.O. è realizzato su architettura uMPS3, un'architettura basata su sei livelli di astrazione. Il secondo livello di Panda+ prende il nome di livello delle code e fornisce l'implementazione delle strutture dati utilizzate dal livello superiore. In particolare, implementa le funzionalità necessarie ai Process Control Block: l'allocazione e la deallocazione dei PCB, la gestione delle code dei PCB, la gestione dell'albero dei PCB, la gestione di un'Active Semaphore Ash che amministra la coda dei PCB bloccati su un semaforo e la gestione dei Namespace dei processi.

2 GESTIONE DI PCB

2.1 ALLOCAZIONE E DEALLOCAZIONE DEI PCB

I PCB possono essere organizzati in code di processi, assimilabili a liste. Ciascuna di queste code è identificata da un elemento sentinella di tipo `list_head`, i cui campi prev e next puntano al primo e all'ultimo elemento della lista. Le liste sono costruite in modo circolare, ossia sono tali per cui il campo prev dell'elemento in testa e il campo next dell'elemento in coda puntino a loro volta alla sentinella. Queste liste sono manipolate e gestite attraverso una serie di macro messe a disposizione dall'ambiente di sviluppo.

Ad ogni PCB è associato un puntatore ad una struttura dati di tipo `pcb_t`. Le funzioni di allocazione e deallocazione dei PCB sono funzioni che permettono rispettivamente di occupare o liberare uno dei PCB appartenenti alla coda dei processi liberi: Occorre inizializzare la lista dei processi liberi, in modo che contenga ciascuno dei 20 PCB dell'array di processi. In particolare, per ogni elemento dell'array si inserisce la sentinella del campo `p_link` della struttura associata al processo in coda alla lista dei semafori liberi.

Quando un PCB viene allocato, viene rimosso l'elemento in testa alla coda dei processi inutilizzati, nel caso in cui essa non sia vuota. L'elemento in testa alla lista corrisponde all'elemento puntato dal campo next della sentinella che punta alla lista. Tutti i campi della struttura dati associata all'elemento vengono inizializzati a 0 o a NULL.

Occorre prestare particolare attenzione alla sentinella del campo `p_list`, che prima puntava alla coda dei processi a cui apparteneva il PCB, ossia alla coda dei processi liberi, e che in seguito all'allocazione deve puntare ad una lista vuota. Per fare in modo che una sentinella punti ad una lista vuota, i suoi campi prev e next devono puntare alla sentinella stessa.

Per deallocare un PCB, il corrispondente processo viene inserito in coda alla coda dei processi liberi. Per fare ciò, si fa in modo che la sentinella del campo `p_list` della struttura punti al primo elemento della lista.

2.2 GESTIONE DELLE CODE DEI PCB

Le funzioni utili alla gestione delle code dei PCB sono tutte quelle funzioni che permettono di creare, gestire e manipolare una coda di processi:

Per creare una coda di processi vuota è sufficiente definire una sentinella di tipo `list_head` i cui campi prev e next puntano alla sentinella stessa. Secondo lo stesso ragionamento, per verificare se ad una coda di processi appartiene almeno un PCB, si verifica che i campi della sentinella non puntino a se stessa.

Per inserire un PCB in coda ad una coda di processi si fa in modo che la sentinella del campo `p_list` della struttura associata punti al primo elemento della lista, ciò equivale a inserire il corrispondente PCB nella lista, prima della sentinella che punta a quella coda di processi.

Per restituire il PCB in testa ad una coda di processi è sufficiente ritornare l'elemento a cui punta il campo next della sentinella della lista. Allo stesso modo, per rimuovere un PCB dalla testa della coda dei processi occorre eliminare l'elemento ottenuto nel medesimo modo, ricollegando opportunamente i concatenatori degli elementi e della sentinella.

2.3 GESTIONE DELL'ALBERO DEI PCB

In aggiunta alla possibilità di partecipare ad una coda di processi, i PCB possono essere organizzati in alberi di processi. Ogni processo genitore contiene una sentinella che punta alla lista dei figli ed ogni processo figlio contiene un puntatore al padre ed una sentinella che punta alla lista dei fratelli, assegnati rispettivamente ai campi `p_child`, `p_parent` e `p_sib` della struttura associata. La sentinella `p_sib` svolge un ruolo simile alla sentinella del campo `p_link`, applicata agli alberi anziché alle code.

Per verificare se la lista dei figli di un PCB contiene almeno un elemento, si controlla se l'elemento sentinella del campo `p_child` della struttura associata al PCB punta ad una lista vuota.

Per inserire un PCB come figlio di un secondo processo, si assegna il secondo processo al campo `p_parent` della struttura che identifica il primo e si inserisce la sentinella del campo `p_sib` del primo PCB nella lista puntata dal campo `p_child` del secondo.

Per eliminare il primo figlio di un PCB si rimuove l'elemento puntato dal campo next della sentinella del campo `p_child` della struttura ad esso associata, che corrisponde all'elemento in testa alla lista dei fratelli a cui appartiene il figlio del PCB.

Per rimuovere un PCB dalla lista dei figli del padre è sufficiente rendersi conto del fatto che la lista dei figli del padre di un PCB corrisponde alla lista dei suoi fratelli. Occorre quindi rimuovere da questa lista l'elemento a cui punta il campo next della sentinella del campo `p_sib` della struttura associata, che corrisponde appunto al processo stesso.

3 GESTIONE DI UN'ASH

L'accesso alle risorse condivise avviene attraverso l'utilizzo di semafori. Ad ogni semaforo è associato un descrittore SEMD con una struttura di tipo `semd_t`. I semafori sono gestiti attraverso una lista dei SEMD liberi o inutilizzati ed una Hash Table, nota come ASH. L'ASH è una struttura dati che contiene i SEMD attivi, indicizzati tramite una chiave. L'Hash Table deve essere dichiarata insieme alla sua dimensione, in questo caso, disponendo al massimo di 20 processi attivi, occorreranno almeno 5 bit per indicizzarli, siccome $2^5 = 32 > 20$. Per la chiave si utilizza la codifica u32.

Occorre inizializzare la lista dei semafori liberi, in modo che contenga ciascuno dei 20 semafori dell'array di SEMD. In particolare, per ogni elemento dell'array si inserisce la sentinella del campo `s_freelink` della struttura associata al SEMD in coda alla lista dei semafori liberi.

Per inserire un PCB nella coda dei processi bloccati associata ad un semaforo, data una chiave, occorre scorrere l'intera ASH per verificare se è presente un SEMD il cui campo `s_key` corrisponde alla chiave cercata. In caso affermativo, si utilizza l'apposita funzione per inserire il PCB nella coda di processi bloccati puntata dalla sentinella del campo `s_procQ` della struttura associata al semaforo cercato. In caso contrario, occorre allocare un nuovo semaforo. Quando un SEMD viene allocato, viene rimosso l'elemento in testa alla lista dei semafori inutilizzati, corrispondente all'elemento puntato dal campo next della sentinella che punta a questa lista. Si inserisce poi nella ASH l'oggetto corrispondente al semaforo, con il descrittore contenuto nel campo `s_link`, associandolo alla chiave fornita. A questo punto, occorre assegnare la chiave all'apposito campo `s_key` del SEMD allocato e inizializzare la sentinella del campo `s_procQ` come una lista vuota, nella cui coda si inserisce la sentinella del campo `p_list` del PCB che punta alla coda dei processi a cui appartiene. Si assegna infine la chiave al campo `s_semAdd` della struttura associata al PCB, che indica il semaforo su cui esso è attualmente bloccato.

Per rimuovere il PCB in testa alla coda dei processi bloccati associata ad un semaforo, data una chiave, occorre scorrere l'intera ASH per verificare se è presente un SEMD il cui campo `s_key` corrisponde alla chiave cercata. Se tale descrittore è presente, si utilizza l'apposita funzione per rimuovere il PCB in testa alla coda dei processi bloccati puntata dal campo `s_procQ` della struttura associata al SEMD cercato. Dopo la rimozione, si verifica se la lista è vuota e, in caso affermativo, si rimuove dall'ASH l'oggetto corrispondente al semaforo, il cui descrittore è contenuto nel campo `s_link` e si aggiunge la sentinella del campo `s_freelink` in coda alla lista dei SEMD liberi.

Per rimuovere un PCB dalla coda dei processi bloccati del semaforo su cui è attualmente bloccato occorre scorrere l'intera ASH per verificare se è presente un SEMD il cui campo `s_key` corrisponde al campo `s_semAdd` della struttura associata al PCB, ossia se corrisponde alla chiave del semaforo su cui il processo è bloccato. Se tale descrittore è presente, si utilizza l'apposita funzione per rimuovere il PCB dalla coda dei processi bloccati puntata dal campo `s_procQ` del SEMD cercato. Dopo la rimozione, si verifica se la lista è vuota e, in caso affermativo, si rimuove dall'ASH l'oggetto corrispondente al semaforo, il cui descrittore

è contenuto nel campo `s_link` e si aggiunge la sentinella del campo `s_freelink` in coda alla lista dei SEMD liberi.

4 GESTIONE DEI NAMESPACE

Possono esistere diverse "visioni" associate ad uno stesso processo, queste prendono il nome di Namespace. Ad ogni Namespace è associato un descrittore NSD con una struttura di tipo `nsd_t`.

Occorre inizializzare una coppia di liste per ogni tipo di Namespace: una per gli NSD liberi o inutilizzati e una per quelli attivi. In questa fase è sufficiente implementare un singolo tipo di Namespace: `NS_PID`. Per ciascuno dei 20 elementi dell'array di NSD si inserisce la sentinella del campo `n_link` della struttura associata al NSD in coda alla lista dei Namespace liberi.

Per ritornare un Namespace associato ad un PCB, dato il suo tipo, occorre scorrere l'array di NSD assegnato al campo namespace della struttura associata al PCB, iterando un numero di volte pari al massimo numero massimo di tipi di NSD implementati. Per ciascuno degli elementi, verifica se al campo `n_type` è associato il tipo cercato e, in caso affermativo, ritorna il corrispondente NSD.

Per associare ad un PCB e a tutti i suoi figli un NSD, dato il suo tipo, occorre scorrere la lista degli NSD liberi e verificare se ad un qualche elemento corrisponde l'NSD specificato. In caso contrario, occorre allocare il Namespace con l'apposito metodo. Per ciascun elemento appartenente alla lista dei figli del PCB puntata dalla sentinella `p_child` della struttura corrispondente, si scorre l'array di NSD associato al campo namespace e nel caso in cui all'elemento considerato sia assegnato NULL o al cui campo `n_type` sia assegnato lo stesso tipo associato al campo `n_type` dell'NSD specificato, si sostituisce l'elemento con questo NSD.

Per allocare un NSD dalla lista corretta, dato il suo tipo, occorre lavorare sulle liste definite appositamente per quel tipo di Namespace. Per operare con tipi di Namespace diversi da `NS_PID`, occorre quindi definire le liste corrispondenti. Si controlla inizialmente che la lista degli NSD liberi non sia vuota e, nel caso, si elimina dalla lista l'elemento in testa. Si inserisce infine la sentinella del campo `n_link` dell'elemento rimosso in coda alla lista degli NSD utilizzati.

Per liberare un NSD reinserendolo nella corretta lista di Namespace liberi, occorre lavorare sulle liste definite appositamente per il tipo assegnato al campo `n_type` del NSD specificato. Si controlla inizialmente che la lista degli NSD attivi non sia vuota e, nel caso, si scorre la lista per verificare se è presente un Namespace che corrisponde a quello fornito. In caso affermativo, si elimina l'NSD dalla lista dei Namespace attivi e si inserisce la sentinella del suo campo `n_link` in coda alla lista degli NSD liberi.

5 MODIFICHE ALLE SPECIFICHE

Per rendere il programma correttamente compilabile è stato necessario sostituire nei file forniti le istruzioni `#include <...>` con `#include "..."`, per permettere al compilatore di trovare i file header.