Relazione progetto Sistemi Operativi

Liberatore Daniele Massa Sandro Palazzo Matteo

Torino, 18 gennaio 2019

NOTE INTRODUTTIVE

Assieme al presente file, vengono fornite due cartelle del progetto, una contenente il codice con informazioni di debug, log-file e documentazione esaustiva, l'altra priva di tali strumenti e con commenti concisi e riassuntivi, rispettivamente `dev` e `release`.

1 COMPILAZIONE

In entrambe le cartelle è presente un `makefile`, a sua volta commentato e corredato di documentazione su come utilizzarlo. Per la compilazione del ramo `release` eseguire il comando **make** e il progetto verrà compilato; per la compilazione del ramo *dev* sono disponibili diverse opzioni di compilazione sotto riassunte.

- **make [...] r:** compilazione di base del progetto. Senza altre opzioni, esegue come la build in `release`, con qualche informazione extra.
- **l:** abilita la generazione di log-file.
- **d:** abilita informazioni di debug generali e critiche in output. Non vengono fornite informazioni di debug sulla generazione dei log-file.
- **s:** ablita solo informazioni di debug critiche in output. Non vengono fornite informazioni di debug generali e sulla generazione dei log-file.
- **g:** abilita informazioni di debug sulla generazione dei log-file in output.
- **clean:** elimina tutti i file eseguibili, oggetto, i log-file e tutte le strutture di IPC create dall'utente. *Utilizzare con cautela.*
- La sola opzione **make** corrisponde al comando **make l s r**, ossia ad una build con generazione di log-file e informazioni di debug critiche.

l, **d**, **s** e **g** sono le opzioni di esecuzione del `makefile`; queste possono essere combinate a piacere. **s** e **d** sono però sono mutuamente esclusive, ossia o si abilita **s** o si abilita **d**. Una volta terminata la compilazione, eseguire `manager.out`.

2 IMPOSTAZIONI

All'interno delle cartelle di entrambi i rami, è presente un file chiamato `opt.conf`: tale file contiene le impostazioni di esecuzione del progetto.
Il file contiene già un esempio di impostazione corretta e una spiegazione delle impostazioni disponibili.
Per completezza, di seguito sarà esposta una breve lista di tali opzioni.

- `G2, G3, G4:` probabilità che uno studente abbia rispettivamente preferenza 2, 3 o 4 come dimensione del gruppo.
- `POP_SIZE:` numero di studenti.
- `NOF_INVITES:` numero di inviti disponibili per studente.
- `MAX_REJECTS:` numero di rifiuti disponibili per studente.
- `SIM_TIME:` durata massima della simulazione.

Ogni opzione ha i suoi vincoli, espressi nel file di configurazione come intervalli.

3 IMPLEMENTAZIONE

3.1 Struttura generale

Il codice può essere scomposto in due macro sezioni, rispettivamente analisi e spedizione degli inviti. Questa divisione è dovuta a una variante dell'applicazione di un algoritmo di sincronizzazione di tipo "lettori-scrittori" dove i primi sono rappresentati dai processi che analizzano gli inviti ricevuti, mentre negli scrittori si identificano quegli studenti che sono in procinto di invitarne altri.
La divisione in "mittenti" e "destinatari" permette, durante l'esecuzione, di avere concorrenza fra studenti appartenenti alla stessa categoria. Per aumentare il grado di multiprogrammazione, le sezioni critiche del singolo studente sono state implementate esclusivamente in quelle porzioni di codice in cui si vanno a modificare delle informazioni potenzialmente utili a tutti i processi in esecuzione.

3.2 Creazione e inizializzazione degli studenti

All'avvio, vengono generate le chiavi per i meccanismi di IPC usati: una coda di messaggi, un segmento di memoria condivisa e un set di 14 semafori (la cui funzione è brevemente descritta all'interno del file `lib.h`). Dopodiché, vengono inizializzati, in memoria condivisa, tre array di lunghezza `POP_SIZE`: uno contiene gli studenti con matricola pari (`even`), il secondo contiene quelli con matricola dispari (`odd`) e il terzo contiene i gruppi creati (`groups`).
Ogni studente è rappresentato da una `struct`, la quale include i seguenti campi: matricola (`regNum`), voto di Architettura degli Elaboratori (`grade`), dimensione del gruppo preferita (`nof_elems`), numero di inviti disponibili (`inviteLeft`), voto finale (`finalGrade`), matricola dell'eventuale capogruppo (`myGroupLeader`, se pari a -1, lo studente non fa parte di alcun gruppo) e una variabile indicante se lo studente in considerazione sta invitando o no (`inviting`).

Anche i singoli gruppi sono rappresentati come struct, i cui campi sono: i membri (`member`, contenuti in una matrice che ha come righe il numero massimo di partecipanti e come colonne matricola e voto di Architettura degli Elaboratori), la grandezza effettiva del gruppo (`size`) e lo stato di quest'ultimo (`isClosed`, chiuso o aperto). Oltre a questi tre array, sono presenti anche altre variabili: una indicante la fine della simulazione causata dal raggiungimento del tempo limite (`timeOut`), altre relative al numero di gruppi ottimali da formare, altre ancora indicanti il numero di gruppi ancora aperti con una data preferenza, altre due contenenti la media dei voti di tutti gli studenti (una per coloro con matricola pari e l'altra per le matricole dispari), quattro variabili per tenere traccia di quanti "mittenti" e di quanti "destinatari" sono attivi contemporaneamente (ai fini della sincronizzazione) e infine, altre quattro per indicare se esattamente uno studente, per ogni preferenza, abbia il permesso di invitare. Tutte queste strutture dati sono identificabili nel file `lib.h`, dove vengono anche accompagnate da una breve descrizione.

3.3 Formazione dei gruppi

L'algoritmo di formazione dei gruppi punta ad associare uno studente, con voto superiore al 66-percentile, a studenti con voto inferiore a questa soglia, al fine di massimizzare la media e il voto finale del gruppo. Per far sì che ciò avvenga in maniera ottimale, e per evitare che diversi mittenti invitino lo stesso studente (consumando inviti inutilmente), solo un certo numero di studenti ha il permesso di spedire inviti contemporaneamente, mentre i restanti aspettano per il proprio turno. La priorità è data alla creazione di gruppi ottimali, ossia gruppi formati da studenti con la stessa preferenza a livello di dimensione di quest'ultimo in modo da evitare una penalizzazione di 3 punti.
Per ogni categoria di studenti esistono due contatori: uno che conta il numero di gruppi ottimali da creare, ed uno che conta il numero corrente di gruppi ottimali aperti. Il primo viene inizializzato con un valore stimato dal processo padre e decrementato ogni qual volta uno studente ottiene il privilegio di essere un capo. Il secondo viene inizializzato a zero ed incrementato ogni qualvolta un gruppo ottimale viene aperto. In questo modo ogni studente saprà il numero massimo di capigruppo possibile, e in base al valore attuale del primo contatore potrà stabilire se diventare un capogruppo ottimale o meno.
Una volta che entrambi i contatori scendono a zero, ossia quando non vi sono più gruppi ottimali da poter formare e quando tutti questi sono stati chiusi; viene concesso ad uno studente alla volta (sempre per evitare lo spreco di inviti, e possibili deadlock) di invitarne altri anche se con una preferenza differente. Un'altra caratteristica di quest'implementazione, è che uno studente scelto per essere un capogruppo, può decidere di voler cedere il suo ruolo, nel momento in cui non dovesse riuscire a formare un gruppo di studenti con la sua stessa preferenza.
Gli studenti continuano a formare gruppi finché la simulazione non termina per via del vincolo temporale o perché non è possibile generare altri gruppi.

3.4 Termine della simulazione

Quando la simulazione giunge al termine per via della chiusura di tutti i gruppi, il processo padre assegna il voto a ogni gruppo che verrà poi stampato dagli studenti.

Se, invece, la simulazione dovesse terminare a causa del raggiungimento del tempo limite, il processo padre notificherebbe tutti gli studenti impostando una data variabile globale con valore logico `TRUE`.
Gli studenti, avranno quindi solamente il permesso di completare potenziali modifiche iniziate su strutture globali, e una volta ultimate quest'ultime usciranno dal ciclo di lavoro principale.
A questo punto si sincronizzeranno con il gestore e gli altri studenti, e una volta ricevuto il voto finale del gestore lo stamperanno a video.
Dopo tutto ciò, gli studenti effettueranno una `exit()`, vengono liberate le aree di memoria dinamicamente allocate e vengono rimossi i componenti d'IPC.