Versione non completa

Struttura del codice:

* masterThread
* collector
* workerThread
* workerThread.h

masterThread : il corpo principale dell’intero progetto, si occupa di:

1. Fare il parsing degli argomenti in input, assegnando alle variabili del sistema I valori adeguati.
2. Creare I threads e assegnargli la funzione definita all’interno di workerThread.c.
3. Creare il processo che eseguira’ il programma collector.
4. Definire le variabili globali esterne dichiarate all’interno di workerThread.h.
5. Raccogliere I nomi dei file e darli in pasto agli workers.
6. Gestire I segnali.

collector: il programma eseguito dopo una fork di masterThread si occupa di raccogliere I risultati dei vari thread worker, ordinarli e stamparli. La comunicazione tra collector e I vari threads worker avviene tramite un socket. Nella comunicazione collector si comporta da master, e gestisce il dialogo direttamente con I vari threads utilizzando un selettore.

workerThread: la funzione worker, funzione assegnata al thread da masterThread, si occupa di connettersi al collector, cosi’ da poter inviare il risultato dei suo calcoli. La funzione fileCalc esegue il calcolo richiesto dal progetto su un file, preso dalla coda condivisa con gli altri thread, coda riempita dal master. La comunicazione avviene tramite la scrittura in caratteri sul socket, traduce anche il long in una stringa. La pulizia della memoria allocata dinamicamente viene affidata a funzioni che vengono pushate nello stack di cleanup del thread e verrano eseguite con politica LIFO all;uscita o cancellazione del Thread stesso (o quando viene chiamata pthread\_cleanup\_pop(1).

workerThread.h: Al fine di garantire la sicurezza nell’utilizzo di dati concorrentemente, worker e master condividono 6 variabili, all’interno della struttura sharedholder definite qui:

* Struct queueEl \* queueHead: puntatore alla testa della list che implementa la coda. Struct queueEl altro non er’ che una stringa (char\*) e un puntatore al prossimo elemento, anche questa definita in questo file.
* Int queueSize: raccoglie la dimensione effettiva della coda, utile per evitare di sforare sia in positivo che in negativo.
* Int masterExitReq: assume valore 0 normalmente, ma viene settata a 0 se viene richiesto ai tthread di finire la coda disponibile e uscire, o 2 per l’uscita immediata.
* Pthread\_mutex\_t mtx: mutex che gestisce l’utilizzo di tutte e 3 le variabili sopracitate, in quanto sono sempre utilizzate in mutua esclusione.
* Pthread\_cond\_t queueNotFull: condizione sul mutex che permette ad un worker di svegliare il master se stava aspettando a cause della coda piena.
* Pthread\_cond\_t queueNotEmpty: condizione sul mutex che permette al master di svegliare uno o piu’ (signal or broadcast) thread se stavano aspettando a causa della coda vuota.