Si chiede di realizzare un programma C, denominato ***farm***, che implementa lo schema di comunicazione tra processi e thread mostrato in Figura 1.

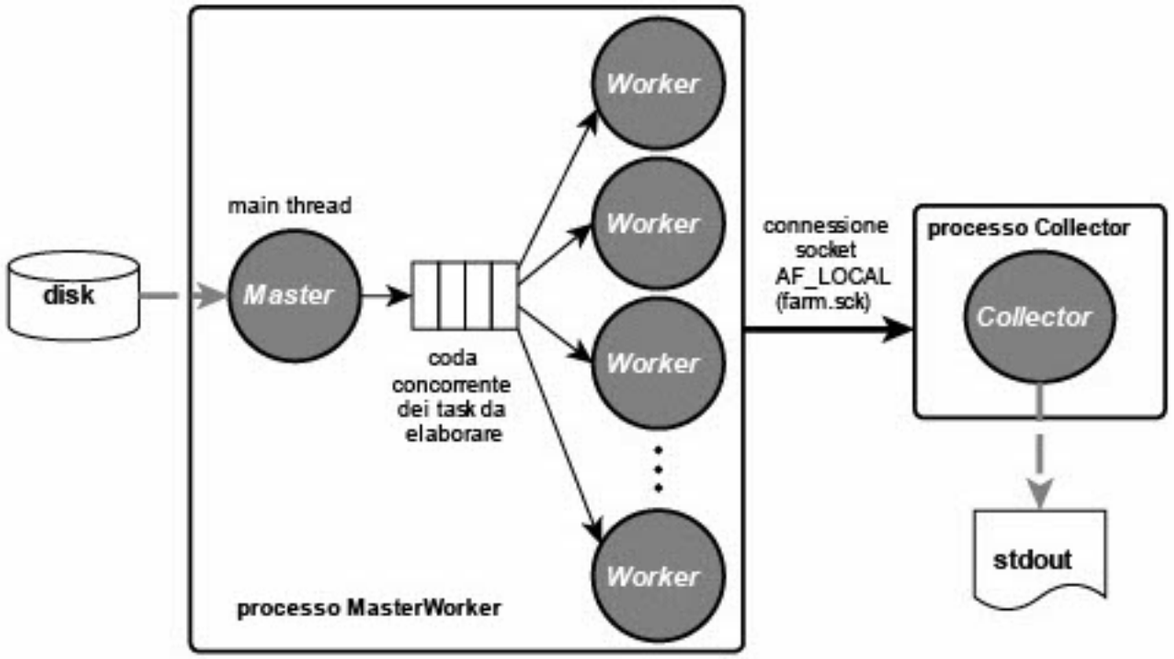

Figura 1 Architettura logica di connessione tra i processi *MasterWorker* e *Collector*

farm è un programma composto da due processi, il primo denominato *MasterWorker* ed il secondo denominato *Collector*. *MasterWorker*, è un processo multi-threaded composto da un thread *Master* e da '*n*' thread *Worker* (il numero di thread Worker può essere variato utilizzando l'argomento opzionale '-n' – vedere nel seguito). Il programma prende come argomenti una lista di file binari contenenti numeri interi lunghi (long) ed un certo numero di argomenti opzionali (opzioni '-n', '-q' e '-t'). Il processo *Collector* viene generato dal processo *MasterWorker*. I due processi comunicano attraverso una connessione socket AF_LOCAL (AF_UNIX). Viene lasciata allo studente la scelta di quale tra i due processi fa da processo master per la connessione socket, così come la scelta se usare una sola connessione o più connessioni socket. Il socket file "***farm.sck***", associato alla connessione AF_LOCAL, deve essere creato all'interno della directory del progetto e deve essere cancellato alla terminazione del programma.

Il processo *MasterWorker* legge gli argomenti passati alla funzione *main* uno alla volta, verificando che siano file regolari, e passa il nome di ogni file (con eventuali altri parametri) ad uno dei thread *Worker* tramite una coda concorrente condivisa (denominata "coda concorrente dei task" in Figura 1). Il generico thread *Worker* si occupa di leggere il contenuto del file ricevuto in input e di fare un calcolo sugli elementi in esso contenuti e quindi di inviare il risultato ottenuto, unitamente al nome del file di input, al processo *Collector* tramite la connessione socket precedentemente stabilita.

Il processo *Collector* attende di ricevere i risultati dai *Worker* e stampa i valori ottenuti sullo standard output nel formato:

risultato nomefile

Il calcolo effettuato dal *Worker* per ogni file ricevuto in ingresso tramite il segmento condiviso è il seguente:

$$result = \sum_{i=0}^{N-1} (i * file[i])$$

dove N è il numero di interi lunghi contenuti nel file e *result* è l'intero lungo che dovrà essere inviato al *Collector*. Ad esempio, supponendo che il file *fileX.dat* passato in input come argomento del *main* abbia dimensione 24 bytes, con il seguente contenuto (si ricorda che i *long* sono codificati con 8 bytes in sistemi Linux a 64bit):

3
2
4

il risultato calcolato dal *Worker* sarà: N=3, $result = \sum_{i=0}^{3-1}(i * file[i]) = (0 * 3 + 1 * 2 + 2 * 4) = 10$.

Gli argomenti che opzionalmente possono essere passati al processo *MasterWorker* sono i seguenti:

- -n <nthread> specifica il numero di thread *Worker* del processo *MasterWorker* (valore di default 4)
- -q <qlen> specifica la lunghezza della coda concorrente tra il thread *Master* ed i thread *Worker* (valore di default 8)
- -t <delay> specifica un tempo in millisecondi che intercorre tra l'invio di due richieste successive ai thread *Worker* da parte del thread *Master* (valore di default 0)

Il processo *MasterWorker* deve gestire i segnali SIGHUP, SIGINT, SIGQUIT, SIGTERM. Alla ricezione di uno di questi segnali il processo deve completare i task eventualmente presenti nella coda dei task da elaborare e quindi terminare dopo aver atteso la terminazione del processo Collector ed effettuato la cancellazione socket file. Il processo Collector maschera i segnali gestiti dal processo *MasterWorker*. Il segnate SIGPIPE deve essere gestito opportunamente dai due processi.

Note

La dimensione dei file in input non è limitata ad un valore specifico. Si supponga che la lunghezza del nome dei file sia non superiore a 255 caratteri.