INTRODUZIONE

In questo progetto utilizziamo 7 file, tra cui 5 file *.c, un makefile* ed un file *header*.

Abbiamo: *gestore.c*, *a.c*, *b.c* e *morte.c*.

È presente anche un file *common.c* che contiene metodi di ausilio a tutti i restanti file.

AVVIO

Per avviare la simulazione digitare il comando *make* da terminale così da compilare automaticamente tutti i file sorgenti e lanciare il gestore.

Per impostare il numero di processi che si vuole immettere nella popolazione e che resterà tale per tutta la durata del procedimento, basta modificare il valore *INIT\_PEOPLE* nell’ header, il quale deve essere >2 per garantire la funzionalità del progetto.

Le variabili *birth\_death* (la quale stabilisce ogni quanti secondi terminare casualmente un processo), *sim\_time* (che stabilisce la durata della simulazione) e *genes* (utile nel calcolo dei genomi), vengono richieste all’utente al momento dell’avvio.

FUNZIONAMENTO

Il tutto ha inizio dal gestore che costituisce il nostro processo padre e che darà vita a tutti gli altri processi.

Esso crea i semafori che serviranno a coordinare lo svolgimento della simulazione, una memoria condivisa (il cui accesso è regolato da un semaforo) visibile a tutti in cui viene inizializzato un array di struct di dimensione *INIT\_PEOPLE* e due code di messaggi, una che verrà utilizzata per le comunicazioni fra i processi A e B ed un’altra che sarà usata dal gestore e i processi B.

Il processo appena nato, tramite una *execlp()* eseguita dalla funzione *nasci* , esegue il proprio codice a seconda del *tipo* e gli si passa come parametro la sua posizione in memoria condivisa.

Il gestore prosegue creando la morte con la funzione *morte(),* la quale non fa altro che eseguire una *fork()* ed una *execlp()* con parametri *birth\_death e sim\_time.*

La morte fa uso della funzione *time()* per avviare un timer che terrà conto del tempo trascorso dall’inizio della simulazione ed in modo ciclico ogni *birth\_death* secondi, tramite una signal SIGUSR1 indirizzata al gestore, comunicherà che dovrà essere terminato un processo con *status* a 0 scelto casualmente.

Sarà poi il gestore in seguito alla ricezione della signal a procedere con la terminazione di un processo, tramite l’ handler *dd\_signal\_handler*.

La morte ha anche il compito di porre fine alla simulazione che avverrà dopo *sim-time* secondi tramite l’invio di una SIGUSR2 al gestore, il quale agirà affinché tutti i processi attivi al momento vengano terminati, compresa la morte stessa.

Dopo la creazione della morte il gestore, tramite l’ausilio di un apposito semaforo, farà partire in modo sincronizzato tutti i processi creati inizialmente, dopodiché non dovrà fare altro che restare in attesa di un messaggio da una B.

Il compito principale dei nuovi processi A e B è esclusivamente quello di scegliersi a vicenda e accoppiarsi sulla base dei criteri specificati nella consegna del progetto.  
Per iniziare la ricerca della A in memoria centrale, la B imposta il proprio status a 1 per segnalare al gestore che non può essere terminata, dopodiché sceglierà la A con genoma più alto tra quelle con *status* a 0.

Una volta scelta la A migliore, B la “blocca” impostandone lo status a 1, in modo da evitare che la stessa venga terminata dal gestore o contattata da altre B.

A questo punto B invia alla A un messaggio con il proprio genoma e PID e resta in attesa di una sua risposta.

In caso di risposta negativa, B imposta gli *status* a 0e si mette alla ricerca di una nuova A, mentre la A abbassa il proprio *target* per facilitare un futuro accoppiamento.

In caso di risposta positiva B manda un messaggio al gestore con il proprio PID e il PID della A e attende di essere terminato.

Il gestore, ricevuti i PID dei processi appena accoppiati, richiama la funzione *accoppiamento\_birth* passandole come parametro i due PID, la quale a sua volta dovrà immettere nella popolazione e alla stessa posizione in memoria condivisa altri due processi con caratteristiche ereditate dai processi genitori appena terminati, tra cui ad esempio il nome che dovrà essere la concatenazione dei nomi dei genitori con in più un’altra lettera a caso tra A-Z.

Dopo *sim\_time* secondi il gestore termina la simulazione, provvedendo all’eliminazione di tutti gli oggetti System V precedentemente creati e di tutti i processi figli, mostrando all’utente un resoconto della simulazione.