# Relazione Progetto

Studente: Sacchetti Lorenzo – 1043821

Insegnamento: Sistemi Operativi

## Struttura del progetto:

Il progetto è suddiviso nelle seguenti parti:

(contenuto del file README)

* Makefile
* README
* bin <- Project main executable
* build <- Static object and intermediate file
* config <- Gonfiguration file for the simulation
* docs <- Documentation
* include <- Header files
* src <- Source files
* tmp <- Temporary file generated by the program

All’interno delle cartelle “include” e “src” sono presenti i file \*.c e i rispettivi header \*.h.

Nelle cartelle “build” e “bin” sono presenti i risultati della compilazione e del linking quindi i file eseguibili.

Nella cartella “config” sono presenti i file \*.txt con le configurazioni. Ogni file presente contiene una configurazione separata.

La compilazione avviene tramite l’uso della make utility in fasi separate. Prima la compilazione con la creazione dei file oggetto, in seguito il linking per la creazione degli eseguibili.

## Funzionamento:

Il progetto viene lanciato tramite l’eseguibile “master.out” che avvia tutti gli altri elementi di controllo della simulazione e, tramite l’uso di un apposito set di semafori, la avvia in accordo con lo stato di prontezza dei vari processi.

Ogni processo all’avvio esegue una funzione di init() che inizializza tutte le strutture dati necessarie come timer e IPC.

Tutti i processi eseguono le loro operazioni ti routine tramite la gestione di segnali lanciati ogni quanto di tempo da appositi timer. Nello specifico il master esegue una stampa delle statistiche ogni secondo, alimentazione genera nuovi atomi ogni STEP secondi, attivatore comanda ACTIVATION\_PER\_SECOND split degli atomi ogni secondo.

La comunicazione di informazioni tra processi avviene principalmente con l’uno di code di messaggi e memoria condivisa. La memoria condivisa è protetta da un semaforo dedicato.

Processo attivatore: questo processo sceglie in maniera casuale uno tra gli atomi disponibili (che non sia una scoria) e, tramite un segnale lo attiva.

Processo atomo: a seguito della fase di init() il processo rimane in pausa fino a quando non riceve un segnale di attivazione. A seguito dei dovuti controlli sul suo stato avvia la fase di split eseguendo una richiesta preventiva al processo inibitore. La richiesta è gestita tramite coda di messaggi. Se il processo inibitore concede lo split, viene generato un altro atomo e il numero atomico opportunamente suddiviso. L’aumento di energia previsto viene salvato in memoria condivisa.

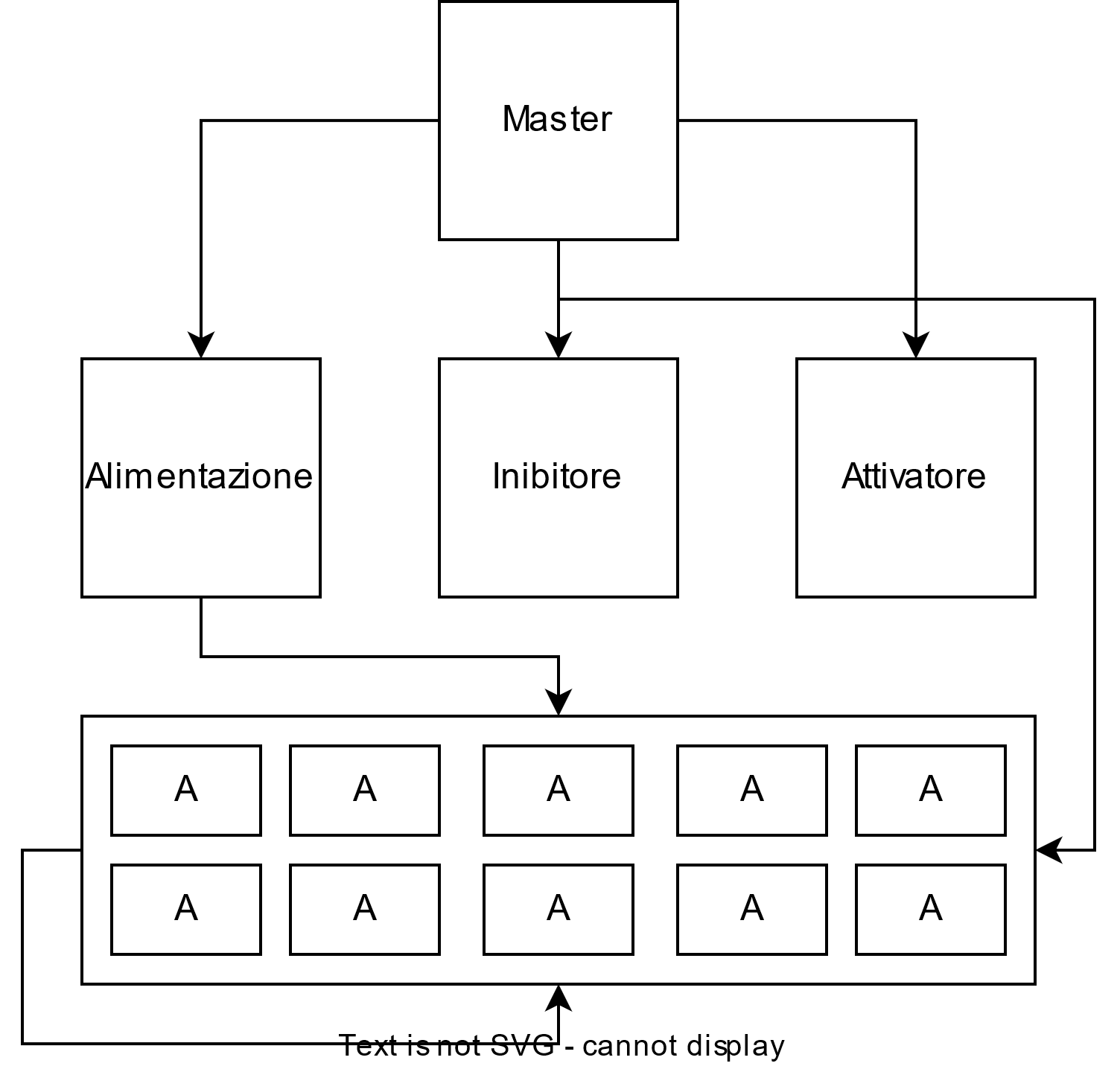
Processo inibitore: a seguito della fase di init() il processo rimane il attesa di ricevere un messaggio da parte di un atomo che richiede di dividersi. Vengono eseguiti i controlli sulle possibili emissioni energetiche e, a seguito dei controlli di tutte le soglie limite, il processo decide di concedere oppure no il permesso all’atomo di dividersi. Tutte le operazioni di comunicazioni avvengono tramite coda di messaggi. I valori energetici complessivi sono mantenuti in memoria condivisa.

Processo alimentazione: genera nuovi atomi eseguendo dei fork.

Processo master: questo processo durante la fase di init() genera tutte le strutture di IPC che in seguito gli altri processi andranno ad utilizzare. Ogni secondo esegue in controlli sui valori della simulazione e se necessario termina in uno dei casi previsti. Ogni secondo stampa i valori parziali e totali della simulazione.

## Fase di creazione dei processi.

I processi vengono creati utilizzando la funzione fork seguita da un’apposita funzione execve. Inizialmente il processo master crea tutti gli elementi per la simulazione. In seguito durante la simulazione gli atomi vengono generati rispettivamente dal processo alimentazione o dagli atomi stessi. Il numero atomico viene comunicato agli atomi tramite una coda di messaggi. Il valore è estratto secondo una distribuzione di probabilità normale che genera numeri nell’intorno di N\_ATOM\_MAX/2. (nella cartella docs è presente il file chart.svg che dimostra che l’estrazione dei numeri avviene secondo una curva normale)



## Fase di scissione.

Casualmente il processo attivatore cerca in memoria condivisa un processo valido per la scissione. Tramite un segnale gli comunica l’ordine di eseguire tale operazione. Il processo atomo esegue una richiesta al processo inibitore che dopo aver effettuato tutte le verifiche di rito decide se consentire la scissione oppure no. La comunicazione tra atomo e inibitore avviene tramite coda di messaggi inviando un’istanza di una struttura dati creata appositamente per condividere le informazioni necessarie alla scissione come pin e numero atomico. In totale la scissione avviene in 3 fasi indicate dalle frecce nello schema sottostante.

