Relazione

Progetto Sistemi Operativi A.A. 2017/2018

Uno schema di coordinamento vagamente ispirato alla Movement Authority per ERTMS/ETCS LV1 e LV2

Componenti

(Data di consegna: 08/06/2018)

- Leonardo Calbi
  - Matricola: 6155786
  - Email: leonardo.calbi@stud.unifi.it
- Alessio Falai
  - Matricola: 6134275
  - Email: alessio.falai@stud.unifi.it
- Tommaso Milli
  - Matricola: 6128367
  - Email: tommaso.milli@stud.unifi.it

Descrizione dettagli implementativi

Il progetto è suddiviso in tre macro funzioni denominate a seconda della modalità di avvio e dello scopo per cui vengono chiamate dal main.

- ***ETCS1***: La funzione *ETCS1* si occupa della completa organizzazione e gestione dell'omonima modalità di avvio, costituendo inizialmente il processo *PADRE_TRENO* definito nelle specifiche. Tale funzione si occupa quindi della creazione e gestione dei singoli *PROCESSI_TRENO*, concorrenti tra loro, in modo da garantire sincronia rispetto alle richieste e mutua esclusione riguardo l'accesso ai file. *ETCS1* si occupa innanzitutto di creare i file testuali per i log e i file legati ai segmenti di ferrovia. Nel vettore *trains* vengono memorizzate informazioni relative a ogni singolo *PROCESSO_TRENO*. In particolare, tali informazioni riguardano il process id (pid) del processo e il suo stato corrente, che può essere EXEC, TERM oppure STOP, ovvero rispettivamente "In esecuzione", "Terminato", "Sospeso". Dopodichè il processo *PADRE_TRENO* passa alla creazione dei vari *PROCESSI_TRENO*, mediante l'utilizzo di *fork*, i quali vengono immediatamente sospesi e spostati in un process group diverso da quello del padre, denominato *children_group*. Tale operazione permette al *PADRE_TRENO* di gestire separatamente e contemporaneamente l'invio di segnali di sincronizzazione ai vari *PROCESSI_TRENO*, mediante l'utilizzo dei segnali SIGCONT e SIGSTOP e delle primitive *raise*, *signal* e *kill*. A questo punto, ogni *PROCESSO_TRENO* riprende l'esecuzione e il codice può essere suddiviso in base ai processi che lo eseguono:

    - *PROCESSO_TRENO*: Inizializza la propria struttura *path*, leggendo da file il proprio percorso, tramite la funzione *read_train_path*. Il treno è pronto per lasciare la stazione di partenza e, dopo aver aggiornato il file di log, si interrompe e aspetta gli altri *PROCESSO_TRENO*. Quando ogni *PROCESSO_TRENO* viene risvegliato dal *PADRE_TRENO*, ciascun *PROCESSO_TRENO* entra nel ciclo legato alla percorrenza dei vari segmenti. Nel ciclo si susseguono le seguenti operazioni: si prova ad accedere al segmento successivo, utilizzando la funzione *toggle_balise*, che ritorna il valore relativo alla possibilità di accesso al segmento che si desidera occupare. A questo punto, il treno aspetta che il segmento venga liberato, se è occupato, oppure lo attraversa utilizzando la funzione *sleep*, come richiesto dal testo. Se il segmento è libero, la funzione *toggle_balise* viene richiamata una seconda volta per garantire l'accesso al precedente segmento ad altri *PROCESSI_TRENO*. Ogni volta che viene effettuata una chiamata alla funzione *toggle_balise*, viene di conseguenza aggiornato il file di log e vengono visualizzati gli accessi ai vari segmenti su console.
    - *PADRE_TRENO*: Aspetta che tutti i figli ancora in esecuzione vengano sospesi, per poi gestire la sincronizzazione del ciclo successivo di richieste, finchè tutti i figli non hanno terminato la loro esecuzione. Tale controllo viene effettuato sfruttando il *process_status* di ogni *PROCESSO_TRENO*, descritto precedentemente.
- ***ETCS2***: La funzione *ETCS2* si occupa di creare il processo *PADRE_TRENO* che gestisce i vari *PROCESSI_TRENO*, ovvero i client del socket *AF_UNIX*. Il principio di funzionamento di questa modalità è molto simile a quello della funzione ETCS1, fatta eccezione per la tecnica di accesso ai vari segmenti, che viene gestita tramite la comunicazione con il server *RBC*. Dunque, ad ogni ciclo di percorrenza, ciascun *PROCESSO_TRENO* richiede di poter accedere al segmento successivo, connettendosi al socket RBC con la funzione *connect* e chiedendo di occupare il segmento desiderato tramite la funzione *socket_auth*, che ritorna il valore relativo all'autorizzazione, concessa o negata, da *RBC*. Inoltre, come richiesto in maniera facoltativa, è stata implementata la gestione del controllo di discordanza tra boe e *RBC*, aggiungendo la gestione mediante file come descritta nella modalità *ETCS1*, così da avere la certezza di una corretta percorrenza dei singoli treni.
- ***ETCS2_RBC***: La funzione *ETCS2_RBC* rappresenta il server del socket *AF_UNIX* che gestisce il corretto susseguirsi di accessi ai segmenti di ferrovia da parte dei *PROCESSI_TRENO*. *RBC* si occupa innanzitutto della creazione del proprio file di log e di due strutture dati che memorizzano lo stato attuale dei segmenti e delle stazioni. Mediante l'utilizzo della funzione *make_named_socket* viene creato il server socket AF_UNIX, denominato "*RBC*", che può accettare al massimo 5 richieste. *RBC* inizializza un vettore di tipo *path*, che contiene i percorsi di ogni *PROCESSO_TRENO*, che riceve dal processo *PADRE_TRENO* tramite socket e inizializza le proprie strutture dati *stations_status* e *balises_status* per tenere traccia delle occupazioni correnti. A questo punto, *RBC* entra in un ciclo, nel quale attende la ricezione di richieste da parte dei *PROCESSI_TRENO*, ognuna delle quali sarà gestita da un processo figlio, generato appositamente.

Tale processo figlio si occuperà di controllare la possibilità di accesso al segmento o alla stazione, comunicarla al *PROCESSO_TRENO* richiedente e aggiornare lo stato delle stazioni e dei segmenti, comunicando con il processo padre tramite l'utilizzo di pipe. Il controllo dell'accesso al segmento viene effettuato tramite la funzione *rbc_auth*, mentre la comunicazione tra *RBC* e i vari *PROCESSI_TRENO* avviene tramite l'utilizzo delle system call *read* e *write*. La scrittura da parte di *ETCS2* e la lettura da parte di *RBC* prevede l'invio e la ricezione della dimensione del messaggio e successivamente dell'effettivo messaggio, per evitare errori di I/O. Quando tutti i treni avranno raggiunto la stazione di arrivo, il socket "*RBC*" viene chiuso e termina l'esecuzione del programma.

Principali strutture dati

- *path*, contiene informazioni relative al percorso di un treno
    - *departure*, contiene l'intero che rappresenta la stazione di partenza
    - *arrival*, contiene l'intero che rappresenta la stazione di arrivo
    - *balises*, è un vettore che contiene la lista dei segmenti e viene inizializzato tramite *malloc* in base al numero di segmenti da attraversare
- process_status, che rappresenta lo stato attuale di un processo e può assumere i valori EXEC, STOP o TERM, che stanno a significare rispettivamente "In esecuzione", "Terminato", "Sospeso".
- *process*, contiene informazioni relative a un singolo processo
    - *pid*, contiene l'id del processo, memorizzato in una variabile intero di tipo *pid_t*
    - *status*, contiene lo stato attuale del processo, memorizzato in una variabile di tipo *process_status* sopra descritta

Principali funzioni

Le principali procedure utilizzate nel programma e richiamate dalle tre macro funzioni sopra descritte sono le seguenti:

- **create_log_files**: La funzione si occupa innanzitutto di creare una cartella denominata "*LOGS*" nello stesso percorso della cartella del progetto e con permessi di tipo 0777. Dopodichè, un ciclo permette la creazione dei vari file di log relativi ai singoli treni, denominati *T1.log*, *T2.log*, *T3.log*, *T4.log* e *T5.log*, ciascuno inserito all'interno della cartella "*LOGS*" precedentemente creata e ognuno generato con permessi 0777. Inoltre, la funzione *open* utilizzata prevede la creazione dei file nel caso in cui non esistano, mentre, nel caso in cui esistano, la cancellazione del loro contenuto, mediante l'utilizzo del flag *O_TRUNC*.
- ***create_max_files***: La funzione si occupa innanzitutto di creare una cartella denominata "*MA1-16*" nello stesso percorso della cartella del progetto e con permessi di tipo 0777. Dopodichè, un ciclo permette la creazione dei vari file relativi allo stato di occupazione di ogni singolo segmento, denominati da *MA1* a *MA16*, ciascuno inserito all'interno della cartella "*MA1-16*" precedentemente creata e ognuno generato con permessi 0777. Inoltre, la funzione *open* utilizzata prevede la creazione dei file nel caso in cui non esistano, mentre, nel caso in cui esistano, la cancellazione del loro contenuto, mediante l'utilizzo del flag *O_TRUNC*.

Dopo la creazione, l'opzione *O_WRONLY* della funzione *open* permette l'apertura di ogni singolo file in modalità sola scrittura; ciò permette di inizializzare ogni segmento con il carattere "*0*", che corrisponde allo stato libero.

- ***create_rbc_log_file***: Come nella funzione *create_log_files* viene creata la cartella "*LOGS*" con le stesse modalità, nel caso in cui non esista, e viene creato il file *RBC.log* con permessi 0777 e inserito all'interno della cartella "*LOGS*" precedentemente creata. L'apertura del file avviene con le stesse modalità descritte nella funzione *create_log_files*.
- ***read_train_path***: La funzione si occupa di leggere il percorso descritto dal descrittore di file passato come parametro e di inserirlo correttamente all'interno di una struttura dati di tipo *path*. Prima di tutto viene inizializzato il vettore *balises* contenuto nella struct di tipo *path*, utilizzando la funzione *malloc* per l'allocazione della memoria e la funzione *memset* per l'inizializzazione al valore "*-1*" di ogni singolo elemento dell'array. Il parsing del percorso viene effettuato carattere per carattere; in particolare, la funzione si occupa di leggere un carattere dal file e, se tale carattere corrisponde a '*S*', allora significa che sta analizzando una stazione, che a seconda dei cicli effettuati viene identificata come quella di partenza o come quella di arrivo, mentre se tale carattere corrisponde ad '*A*', allora significa che sta analizzando un segmento. Nel caso in cui venga analizzato un segmento, la funzione si occupa di ricostruire il valore intero associato ai caratteri che descrivono le varie boe. Queste operazioni di analisi vengono ripetute fino a che è possibile leggere caratteri dal file, ovvero fino a che non si raggiunge l'*EOF*, per poi ritornare il completo percorso del treno, correttamente inserito nella struttura dati di tipo *path*.
- ***update_log***: La funzione si occupa della scrittura all'interno del file di log del treno che viene indicato dal parametro intero *index*. In particolare, alla funzione vengono passati i due parametri stringa *current* e *next*, che permettono di identificare il segmento attuale o la stazione di partenza e il prossimo segmento o la stazione di arrivo del treno. Dunque, la funzione procede con l'apertura in modalità di sola scrittura *O_WRONLY* del file di log associato al treno *index*. Dopodichè, nel file viene effettuata una scrittura che comprende la stringa *current*, la stringa *next* e una stringa che rappresenta la data e l'ora dell'operazione, generata utilizzando le funzioni *time*, *localtime* e *strftime*. La corretta rappresentazione della data e dell'ora secondo il formato locale italiano viene effettuata grazie alla funzione *setlocale*, chiamata direttamente nel *main*. Inoltre, l'apertura del file tramite flag *O_APPEND* consente di scrivere in fondo al file, evitando di sovrascrivere.
- ***toggle_balise***: La funzione prende in ingresso i parametri *balise* e *w* e si occupa innanzitutto di aprire il file relativo all'intero *balise* sia in lettura che scrittura, tramite il flag *O_RDWR* e di effettuare il lock di scrittura su tale file, mediante la funzione *fcntl*, con flag *SET_LK* (Lock non bloccante). Dopodichè, se il lock è andato a buon fine, viene letto il carattere presente nel file relativo all'intero *balise*, mentre se il lock dovesse fallire, cioè nel caso in cui un altro treno stesse modificando il file relativo allo stesso segmento, allora verrebbe stampato il relativo ERRNO a scopo informativo. Se il carattere letto è diverso dal carattere *w* passato come parametro, allora è possibile scrivere *w* sul file.

Infine, dopo aver effettuato l'unlock del file, viene ritornato il valore intero "*1*" se l'operazione di scrittura è andata a buon fine, "*0*" altrimenti. Il carattere in ingresso *w* può contenere i valori "*0*" o "*1*", che indicano rispettivamente la volontà di liberare il segmento *balise* o di occuparlo.

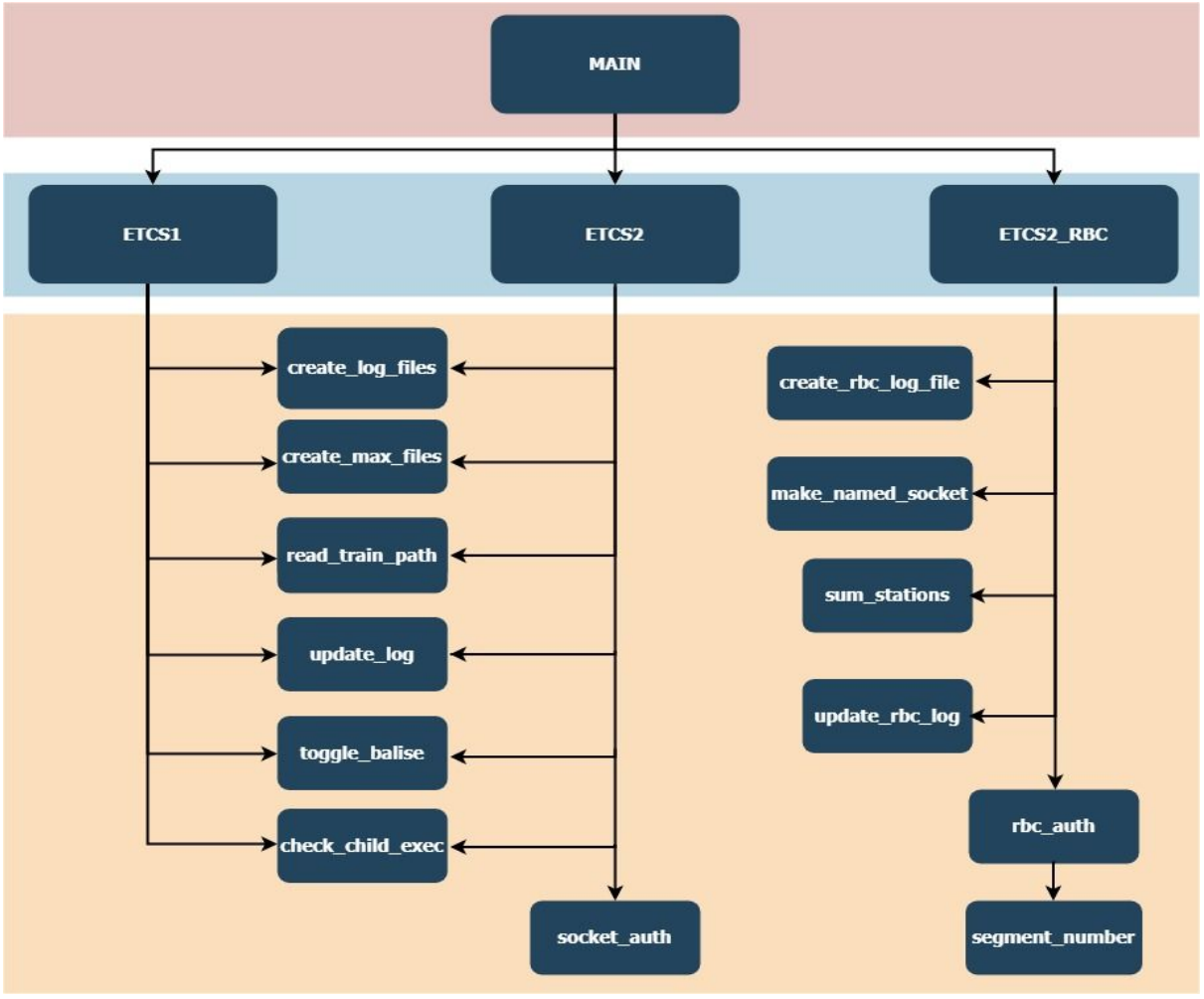
- ***check_child_exec***: La funzione controlla che lo stato di ogni *PROCESSO_TRENO* sia "Terminato". Se ciò avviene viene ritornato il valore "*1*", altrimenti viene ritornato il valore "*0*".
- ***socket_auth***: La funzione permette la comunicazione con il server RBC da parte del *PROCESSO_TRENO* identificato dal parametro intero *index*. In particolare, tale comunicazione prevede la scrittura del parametro *index*, delle due stringhe *current* e *next*, che identificano rispettivamente il segmento attuale e quello successivo, previa scrittura della dimensione di ognuno. Dopodichè, la funzione aspetta una risposta dal server RBC, utilizzando una *read* bloccante, che gli comunica l'esito dell'autorizzazione. Infine, viene ritornato l'esito acquisito.
- ***update_rbc_log***: La funzione si occupa della scrittura all'interno del file di log del server RBC, delle informazioni relative allo stato del treno indicato dal parametro intero *index*. In particolare, alla funzione vengono passati i due parametri stringa *current* e *next*, che permettono di identificare il segmento attuale o la stazione di partenza e il prossimo segmento o la stazione di arrivo del treno. Dunque, la funzione procede con l'apertura in modalità di sola scrittura *O_WRONLY* del file di log associato al server RBC. Dopodichè, nel file viene effettuata una scrittura che comprende l'indice del treno richiedente autorizzazione, la stringa *current*, la stringa *next*, l'autorizzazione concessa o meno e una stringa che rappresenta la data e l'ora dell'operazione, generata utilizzando le funzioni *time*, *localtime* e *strftime*. La corretta rappresentazione della data e dell'ora secondo il formato locale italiano viene effettuata grazie alla funzione *setlocale*, chiamata direttamente nel *main*. Inoltre, l'apertura del file tramite flag *O_APPEND* consente di scrivere in fondo al file, evitando di sovrascrivere.
- ***make_named_socket***: La funzione si occupa della creazione di un server socket di nome *filename*, passato come parametro, di tipo *AF_UNIX*, *SOCK_STREAM* e con protocollo di default, tramite l'utilizzo delle funzioni *socket* e *bind*. Nel caso in cui si verifichino degli errori il programma stamperà le stringhe "*Socket init*" o "*Socket bind*" e i relativi *ERRNO*, per poi uscire con stato *EXIT_FAILURE*. Nel caso in cui la creazione vada a buon fine viene ritornato il file descriptor del server socket creato.
- ***sum_stations***: La funzione controlla che ogni *PROCESSO_TRENO* sia arrivato alla propria stazione di destinazione, utilizzando la struttura dati *stations_status* passata come parametro. In particolare, questa operazione viene effettuata contando il numero di treni che attualmente si trovano in una stazione e ritornando tale somma, che verrà poi utilizzata dal server RBC per determinare se continuare l'esecuzione.
- ***rbc_auth***: La funzione si occupa di aggiornare lo stato delle strutture dati *balises_status* e *stations_status* del server RBC, in base ai parametri stringa *current* e *next*, che indicano rispettivamente il segmento attuale e quello successivo di un particolare *PROCESSO_TRENO*.

Nel caso in cui venga effettuato un passaggio fra una stazione e un segmento oppure un passaggio fra un segmento e un altro segmento oppure un passaggio fra un segmento e una stazione, vengono controllate le posizioni relative a tali accessi nelle strutture dati del server RBC e di conseguenza modificate se consentito, ovvero se i segmenti successivi sono liberi. La funzione ritorna il valore "*1*" per un corretto aggiornamento, mentre ritorna il valore "*0*" in caso di errore.

- ***segment_number***: La funzione si occupa di determinare la sottostringa che inizia dalla prima occorrenza del carattere *c* passato come parametro, fino alla fine della stringa, anch'essa passata come parametro. Tale sottostringa viene poi convertita in un intero, tramite la funzione *atoi*, che viene ritornato. Lo scopo di questa funzione è quello di determinare l'intero contenuto in una stringa del tipo "*MA12*" o "*S4*". In tali esempi la funzione dovrebbe essere richiamata passando come parametri la stringa e il carattere '*A*' oppure '*S*' e la funzione ritornerebbe rispettivamente i valori interi "*12*" e "*4*". In caso di errore la funzione ritorna il valore "*-1*".
- ***interrupt_children***: La funzione è un handler per il segnale *SIGINT*, che si occupa di propagare tale segnale anche al gruppo dei *PROCESSI_TRENO*, denominato *children_group*. Tale handler è necessario per garantire la terminazione dei figli nel caso in cui venga terminato il processo padre, ad esempio tramite console.

Albero delle chiamate di funzioni

- *main*
  - *interrupt_children*
  - *ETCS1*
    - *create_log_files*
    - *create_max_files*
    - *read_train_path*
    - *update_log*
    - *toggle_balise*
    - *check_child_exec*
  - *ETCS2*
    - *create_log_files*
    - *create_max_files*
    - *read_train_path*
    - *socket_auth*
    - *toggle_balise*
    - *update_log*
    - *check_child_exec*
  - *ETCS2_RBC*
    - *create_rbc_log_file*
    - *make_named_socket*
    - *sum_stations*
    - *rbc_auth*
      - *segment_number*
    - *update_rbc_log*



Verifica correttezza rispetto alle specifiche

ETCS1

In questa esecuzione andiamo a descrivere lo stato dei treni T2 e T3, che potrebbero “scontrarsi” nel segmento “MA12”.

T3.log

Mostra Adesso Cancella Ricarica Condividi Cerca

```
[Attuale: S4], [Next: MA14], 08 Giugno 2018 15:32:46
[Attuale: MA14], [Next: MA15], 08 Giugno 2018 15:32:49
[Attuale: MA15], [Next: MA16], 08 Giugno 2018 15:32:52
[Attuale: MA16], [Next: MA12], 08 Giugno 2018 15:32:55
[Attuale: MA12], [Next: S8], 08 Giugno 2018 15:32:58
```

T2.log

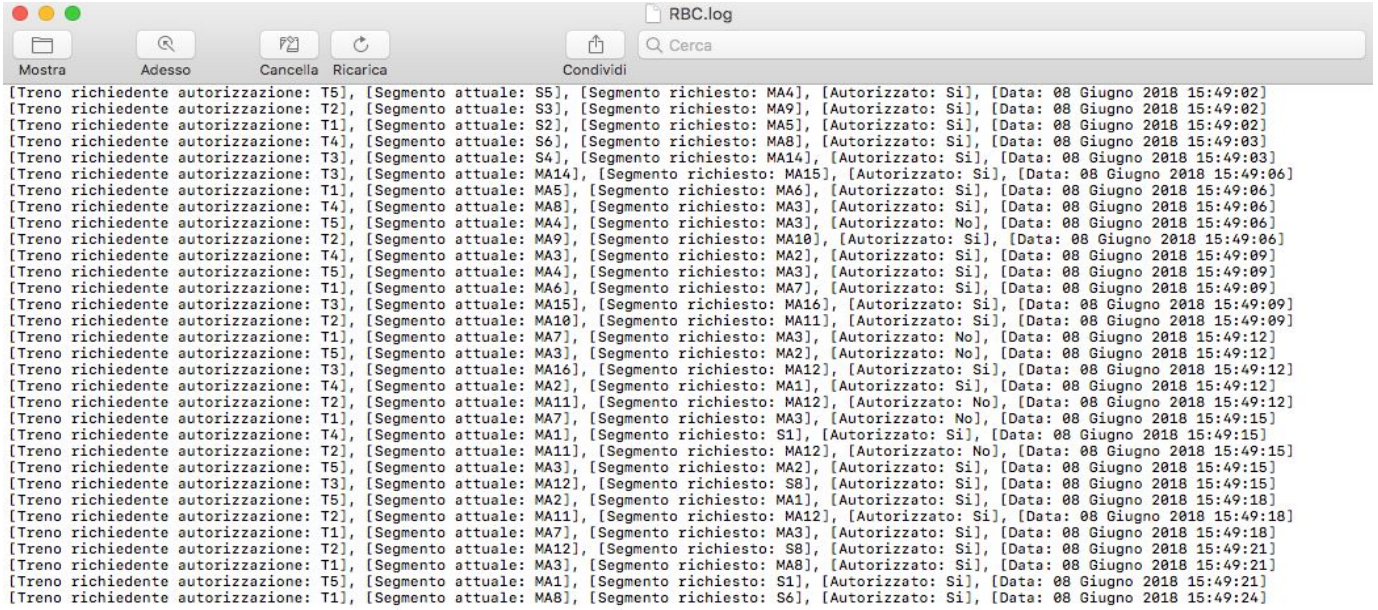
Mostra Adesso Cancella Ricarica Condividi Cerca

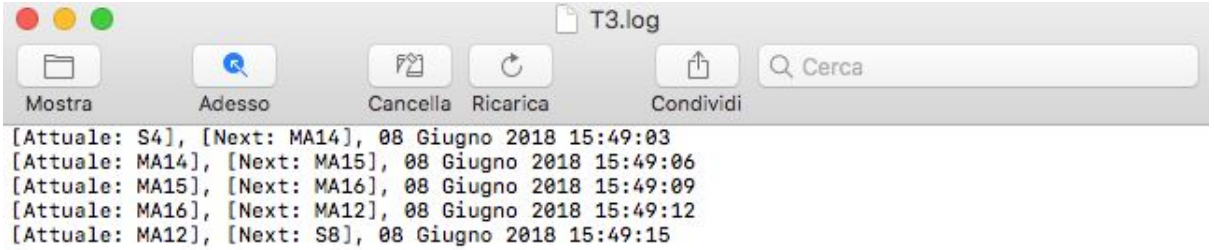
```
[Attuale: S3], [Next: MA9], 08 Giugno 2018 15:32:46
[Attuale: MA9], [Next: MA10], 08 Giugno 2018 15:32:49
[Attuale: MA10], [Next: MA11], 08 Giugno 2018 15:32:52
[Attuale: MA11], [Next: MA12], 08 Giugno 2018 15:32:55
[Attuale: MA11], [Next: MA12], 08 Giugno 2018 15:32:58
[Attuale: MA12], [Next: S8], 08 Giugno 2018 15:33:01
```

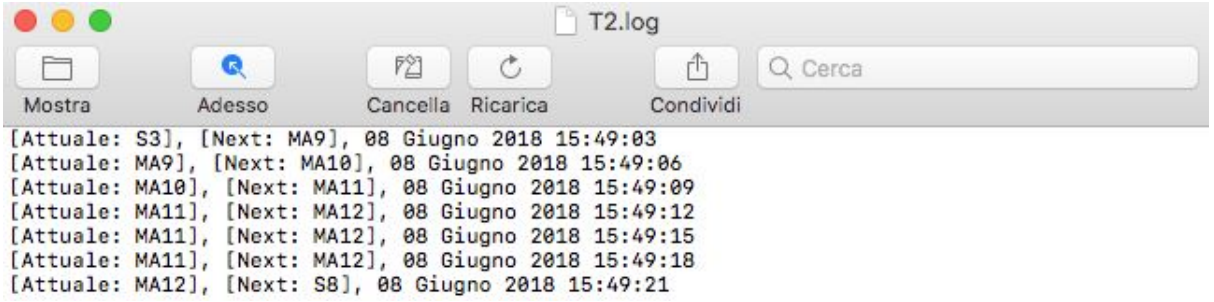
Dall'analisi dei due file di log possiamo vedere che i treni procedono normalmente e avanzano nel loro percorso, finchè non si incontrano nel segmento "MA12". Alla quarta riga del log del treno T3 e del treno T2 possiamo vedere che entrambi i treni richiedono l'accesso allo stesso segmento nello stesso istante di tempo, ovvero alle ore *15:32:55*. Dalla lettura delle righe successive si evince che l'accesso è stato autorizzato solamente al treno T3; mentre T2 è costretto ad attendere altri 3 secondi di tempo per ritentare l'accesso. Il secondo tentativo di accesso del treno T2 è descritto dalla quinta riga del log, nella quale si ripresentano gli stessi segmenti, attuale e successivo, della quarta riga. Alle ore *15:32:58* viene quindi consentito l'accesso al segmento "MA12" anche al treno T2, dato che nello stesso istante il treno T3 accede alla stazione di arrivo "S8".

ETCS2 / ETCS2 RBC

In questa esecuzione andiamo a descrivere lo stato dei treni T2 e T3, che potrebbero "scontrarsi" nel segmento "MA12".







Dall'analisi dei tre file di log possiamo vedere che i treni procedono normalmente e avanzano nel loro percorso, poiché il server RBC autorizza ogni richiesta effettuata, finchè non si incontrano nel segmento "MA12". Alla quarta riga del log del treno T3, che corrisponde alla diciottesima riga del log di RBC, e alla quarta riga del log del treno T2, che corrisponde alla ventesima riga del log di RBC, possiamo vedere che entrambi i treni richiedono l'accesso allo stesso segmento nello stesso istante di tempo, ovvero alle ore *15:49:12*. Analizzando l'ordine delle righe del file di log di RBC possiamo notare che l'autorizzazione all'accesso al segmento "MA12" viene garantita al treno T3 in quanto la sua richiesta viene analizzata prima di quella del treno T2. Dalla lettura delle righe successive si evince che l'accesso è stato autorizzato solamente al treno T3; mentre T2 è costretto ad attendere altri 3 secondi di tempo per ritentare l'accesso. Il secondo tentativo di accesso del treno T2 è descritto dalla quinta riga del log, nella quale si ripresentano gli stessi segmenti, attuale e successivo, della quarta riga. Anche il secondo tentativo non permette al treno T2 di accedere al segmento "MA12" per lo stesso motivo, ovvero perché la richiesta di T2 arriva prima che T3 abbia liberato il segmento (Riga 23, per il treno T2, e riga 25, per il treno T3, del file di log RBC). Alle ore *15:49:18* viene effettuato il terzo tentativo, che consente l'accesso al segmento "MA12" anche al treno T2, dato che alle ore *15:49:15* il treno T3 accede alla stazione di arrivo "S8", liberando il segmento "MA12".

Istruzioni per l'avvio

Per avviare il programma è necessario aprire un terminale e spostarsi nella cartella del progetto tramite il comando *cd*. Dopodichè, se nella cartella non è presente il file eseguibile *main*, è possibile crearlo utilizzando il comando *gcc -o main main.c*. A questo punto, è possibile scegliere una delle seguenti modalità di avvio:

- **ETCS1**: Una volta creato l'eseguibile è possibile avviarlo, con la modalità ETCS1, utilizzando il comando *./main ETCS1*. L'esecuzione prevede la stampa su terminale degli aggiornamenti di percorrenza dei vari treni.
- **ETCS2**: Come per ETCS1, dopo aver creato l'eseguibile è possibile avviare il programma con la modalità ETCS2 utilizzando il comando *./main ETCS2*. Il programma aspetterà l'apertura del server RBC, per poi stampare su terminale gli aggiornamenti di percorrenza dei vari treni.
- **ETCS2 RBC**: Dopo aver avviato il programma con la modalità ETCS2 è necessario aprire una seconda finestra di terminale, spostarsi nella cartella del progetto tramite il comando *cd* e avviare il programma con la modalità ETCS2 RBC, utilizzando il comando *./main ETCS2 RBC*. Questa modalità garantisce il funzionamento del programma soltanto se precedentemente era stato avviato il software con l'opzione ETCS2 da un'altra finestra di terminale. L'esecuzione prevede la stampa su terminale delle richieste di accesso ai vari segmenti da parte di ogni treno.

NOTE: Nel caso in cui, dopo l'avvio tramite il comando *./main*, non venga specificato alcun argomento, oppure l'argomento specificato non corrisponde alle stringhe sopra descritte, il programma stamperà su terminale la stringa di errore "*Argument error*".