Relazione progetto Sistemi Operativi 17/18

Data di consegna: XX/XX/XXXX

Una copia completa del progetto è reperibile sotto GPL al seguente indirizzo:
https://github.com/Errore418/ProgettoSO1718

Sommario

Autori

- Nave Claudio, 6101907, claudio.nave@stud.unifi.it
- Scaringella Andrea, 6143224, andrea.scaringella@stud.unifi.it

Architettura software

Il primo programma ad essere eseguito è *ertms*: tale processo è responsabile di preparare tutti i file e le cartelle necessarie ai processi futuri. È deputato inoltre alla preparazione e condivisione delle variabili condivise. Può essere eseguito solo con una combinazione limitata di argomenti: *etcs1*, *etcs2*, *etcs2 rbc*. Nei primi due casi *ertms* avvierà il processo *train* un numero di volte pari a quello specificato dalla macro *NUMBER_OF_TRAINS*, passandogli di volta in volta il suo numero identificativo (un numero intero compreso tra 1 e *NUMBER_OF_TRAINS*) e la rispettiva modalità di avvio. Nel terzo caso, invece, *ertms* avvierà il processo *rbc*, passandogli tramite comunicazione socket tutti i percorsi dei treni (contenuti in file

csv localizzati in *resources/routes*). Sono presenti inoltre diversi moduli di utilità: *log.c* racchiude funzioni per la formattazione e scrittura dei file di log, *route.c* contiene funzioni per la lettura, decodifica e gestione dei percorsi dei treni, *common.c* risulta, invece, un contenitore con funzioni di uso generale fattorizzate in un unico punto per evitare ridondanza di codice. Per una maggiore agevolezza nello sviluppo è stato completamente separato il codice delle funzioni da tutto il resto, quest'ultimo inserito nel corrispettivo *file header*. Appositi *include guard* evitano inclusioni ripetute.

Descrizione funzionamento

Il programma prevede due modalità di avvio. Tali modalità si distinguono tra le altre cose per come i processi treni chiedono l'autorizzazione ad avanzare. Di seguito una trattazione ad alto livello del flusso di esecuzione di entrambe le modalità.

Modalità ETCS1

La modalità ETCS1 necessita di un solo terminale in cui *ertms* viene lanciato con opzione *etcs1*. Prima di avviare i treni, *ertms* crea le cartelle *resources/MAx* e *resources/log*. Crea inoltre un numero di file *MA* pari a quello specificato dalla macro *NUMBER_OF_MA* inizializzandoli a 0. Inizializza e mette in condivisione una struttura dati contenente alcune variabili che verranno usate dai treni. Se tutte queste operazioni hanno avuto successo, avvia i treni e si sospende aspettandone la terminazione. Quando un treno viene avviato provvede come prima cosa a ricavarsi un puntatore alle variabili condivise predisposte da *ertms*, a registrare il proprio numero identificativo, a settare alcuni puntatori a funzione a seconda della modalità di avvio e a leggere il proprio percorso e immagazzinarlo in una lista. Dopodiché il treno può cominciare a muoversi per completare il suo percorso. Per muoversi un treno effettua un ciclo composto da tre parti fondamentali: lock della posizione corrente, richiesta di spostamento, movimento. Ogni ciclo viene iniziato solo quando tutti i treni sono pronti. In *etcs1* il lock della posizione corrisponde a fissare un lock sul file *MA* corrispondente all'attuale posizione del treno. Tale operazione si conclude solo quando viene correttamente completata da tutti i treni. La richiesta di spostamento, invece, risulta essere una lettura del file *MA* in cui il treno vorrebbe spostarsi. Se tale file contiene uno 0 la richiesta è autorizzata, al contrario il segmento risulta essere già occupato da un treno e perciò la richiesta è negata. Prima di poter leggere un file ogni treno deve riuscire a conquistare il corrispettivo lock, ciò al tempo stesso evita che due treni si spostino sullo stesso file e permette a un treno di attendere l'eventuale liberazione di un segmento. Lo spostamento (uguale per entrambe le modalità) viene eseguito solo se l'autorizzazione è stata concessa e consiste nello scrivere 0 nel *MA* da cui ci si sposta e 1 in quello di arrivo.

Prima di terminare un ciclo e mettersi in sospensione ogni treno chiude i file aperti, ciò provoca il rilascio dei lock associati e il corrispettivo sblocco di eventuali treni in attesa.

Modalità ETCS2

La modalità ETCS2 necessita di due terminali: uno in cui *ertms* viene lanciato con opzione *etcs2* e nell'altro con opzione *etcs2 rbc*. Nel primo terminale *ertms* svolge le stesse funzioni già descritte precedentemente. Un treno avviato in modalità *etcs2* tenta di connettersi al socket generato dal processo *rbc*. Il flusso di esecuzione è pressoché simile a quello precedente, con alcune differenze: il lock della posizione corrente corrisponde alla notifica ad *rbc* della posizione occupata dal treno. Mentre la richiesta di movimento corrisponde all'invio di un messaggio a *rbc* con relativa attesa della sua risposta. Nel secondo terminale *ertms* non crea nessuna cartella o file, ma predispone semplicemente le variabili condivise per *rbc*. Dopo aver avviato *rbc*, si connette al suo socket e gli invia tutti i percorsi dei treni. Da parte sua *rbc* una volta avviato predispone il socket per la comunicazione con *ertms* e successivamente quello per la comunicazione con i treni, mettendosi in attesa delle loro richieste di connessione. Una volta accettata la connessione di un treno *rbc* genera un processo figlio per gestire le sue richieste. I processi figli di *rbc* avviano un ciclo infinito composto da due parti: ricezione della posizione attuale del treno e ricezione della richiesta di movimento del treno. Il ciclo viene interrotto solo quando un treno chiude la connessione, cioè una volta arrivato a destinazione. La ricezione della posizione di un treno consiste nella lettura del numero inviato e nel lock del corrispettivo mutex (tali mutex sono contenuti in un array predisposto da *ertms* e corrispondono ai file *MA*). La ricezione della richiesta di movimento di un treno consiste nella lettura dei dati inviati da un treno (numero di treno, posizione attuale e posizione richiesta), nel lock del mutex corrispondente alla posizione richiesta dal treno, alla lettura dello stato del segmento, all'eventuale aggiornamento della posizione dei treni e alla risposta affermativa o positiva. Lo stato dei segmenti è rappresentato sotto forma di un array di interi con la stessa codifica dei file *MA* (0 per un segmento libero, 1 per uno occupato). Al termine del ciclo vengono rilasciati i mutex posseduti, provocando lo sblocco di eventuali processi in attesa e dei rispettivi treni attendenti una risposta.

Analisi log

Dall'analisi dei log si possono osservare tutti i benefici derivanti dalla sincronizzazione estrema tra i processi. Dal momento che ogni ciclo viene effettuato insieme da tutti i treni, le righe dei vari log dei treni corrispondono a tali cicli. Ciò agevola di molto la verifica di eventuali errori. Ad esempio per verificare che in ogni segmento soggiorni al più un solo treno basterà controllare che le

posizioni attuali dei treni in una determinata riga siano tutte diverse tra loro. L'uso dei lock, invece, consente un movimento ottimizzato: un treno che volesse muoversi in un segmento occupato aspetterà che il treno davanti si sposti e che rimuova i lock, sbloccandolo e facendogli trovare il segmento libero, riuscendo perciò a muoversi nello stesso ciclo. Di seguito i cinque log dei treni avviati in modalità ETCS1:

- T1.log

```
[Attuale: S2], [Prossimo: MA5], 12 August 2018 17:42:20
[Attuale: MA5], [Prossimo: MA6], 12 August 2018 17:42:23
[Attuale: MA6], [Prossimo: MA7], 12 August 2018 17:42:26
[Attuale: MA7], [Prossimo: MA3], 12 August 2018 17:42:30
[Attuale: MA3], [Prossimo: MA8], 12 August 2018 17:42:33
[Attuale: MA8], [Prossimo: S6], 12 August 2018 17:42:36
[Attuale: S6], [Prossimo: ---], 12 August 2018 17:42:39
```

- T2.log

```
[Attuale: S3], [Prossimo: MA9], 12 August 2018 17:42:20
[Attuale: MA9], [Prossimo: MA10], 12 August 2018 17:42:23
[Attuale: MA10], [Prossimo: MA11], 12 August 2018 17:42:26
[Attuale: MA11], [Prossimo: MA12], 12 August 2018 17:42:30
[Attuale: MA11], [Prossimo: MA12], 12 August 2018 17:42:33
[Attuale: MA12], [Prossimo: S8], 12 August 2018 17:42:36
[Attuale: S8], [Prossimo: ---], 12 August 2018 17:42:39
```

- T3.log

```
[Attuale: S4], [Prossimo: MA14], 12 August 2018 17:42:20
[Attuale: MA14], [Prossimo: MA15], 12 August 2018 17:42:23
[Attuale: MA15], [Prossimo: MA16], 12 August 2018 17:42:26
[Attuale: MA16], [Prossimo: MA12], 12 August 2018 17:42:30
[Attuale: MA12], [Prossimo: S8], 12 August 2018 17:42:33
[Attuale: S8], [Prossimo: ---], 12 August 2018 17:42:36
```

- T4.log

```
[Attuale: S6], [Prossimo: MA8], 12 August 2018 17:42:20
[Attuale: MA8], [Prossimo: MA3], 12 August 2018 17:42:23
[Attuale: MA8], [Prossimo: MA3], 12 August 2018 17:42:26
[Attuale: MA3], [Prossimo: MA2], 12 August 2018 17:42:30
[Attuale: MA2], [Prossimo: MA1], 12 August 2018 17:42:33
[Attuale: MA1], [Prossimo: S1], 12 August 2018 17:42:36
[Attuale: S1], [Prossimo: ---], 12 August 2018 17:42:39
```

- T5.log

```
[Attuale: S5], [Prossimo: MA4], 12 August 2018 17:42:20
[Attuale: MA4], [Prossimo: MA3], 12 August 2018 17:42:23
[Attuale: MA3], [Prossimo: MA2], 12 August 2018 17:42:26
[Attuale: MA2], [Prossimo: MA1], 12 August 2018 17:42:30
[Attuale: MA1], [Prossimo: S1], 12 August 2018 17:42:33
[Attuale: S1], [Prossimo: ---], 12 August 2018 17:42:36
```

Come si può osservare nessun treno si sposta su un segmento già occupato. T2 e T4, infatti, rimangono fermi per far passare rispettivamente T3 e T5. Del resto quando questi secondi si spostano vengono seguiti immediatamente dai primi. Tale movimento ottimizzato riduce drasticamente le volte in cui ad un treno viene negata l'autorizzazione a spostarsi. Ciò è particolarmente evidente dal log di *rbc* e dei treni avviati in modalità ETCS2:

- RBC.log

```
[Treno richiedente autorizzazione: T5], [Segmento attuale: S5], [Segmento richiesto: MA4], [Autorizzato: SI], [Data: 12 August 2018 17:44:43]
[Treno richiedente autorizzazione: T1], [Segmento attuale: S2], [Segmento richiesto: MA5], [Autorizzato: SI], [Data: 12 August 2018 17:44:43]
[Treno richiedente autorizzazione: T4], [Segmento attuale: S6], [Segmento richiesto: MA8], [Autorizzato: SI], [Data: 12 August 2018 17:44:43]
[Treno richiedente autorizzazione: T2], [Segmento attuale: S3], [Segmento richiesto: MA9], [Autorizzato: SI], [Data: 12 August 2018 17:44:43]
[Treno richiedente autorizzazione: T3], [Segmento attuale: S4], [Segmento richiesto: MA14], [Autorizzato: SI], [Data: 12 August 2018 17:44:43]
[Treno richiedente autorizzazione: T1], [Segmento attuale: MA5], [Segmento richiesto: MA6], [Autorizzato: SI], [Data: 12 August 2018 17:44:47]
[Treno richiedente autorizzazione: T4], [Segmento attuale: MA8], [Segmento richiesto: MA3], [Autorizzato: SI], [Data: 12 August 2018 17:44:47]
[Treno richiedente autorizzazione: T2], [Segmento attuale: MA9], [Segmento richiesto: MA10], [Autorizzato: SI], [Data: 12 August 2018 17:44:47]
[Treno richiedente autorizzazione: T3], [Segmento attuale: MA14], [Segmento richiesto: MA15], [Autorizzato: SI], [Data: 12 August 2018 17:44:47]
[Treno richiedente autorizzazione: T5], [Segmento attuale: MA4], [Segmento richiesto: MA3], [Autorizzato: NO], [Data: 12 August 2018 17:44:47]
[Treno richiedente autorizzazione: T3], [Segmento attuale: MA15], [Segmento richiesto: MA16], [Autorizzato: SI], [Data: 12 August 2018 17:44:50]
[Treno richiedente autorizzazione: T4], [Segmento attuale: MA3], [Segmento richiesto: MA2], [Autorizzato: SI], [Data: 12 August 2018 17:44:50]
[Treno richiedente autorizzazione: T1], [Segmento attuale: MA6], [Segmento richiesto: MA7], [Autorizzato: SI], [Data: 12 August 2018 17:44:50]
[Treno richiedente autorizzazione: T2], [Segmento attuale: MA10], [Segmento richiesto: MA11], [Autorizzato: SI], [Data: 12 August 2018 17:44:50]
[Treno richiedente autorizzazione: T5], [Segmento attuale: MA4], [Segmento richiesto: MA3], [Autorizzato: SI], [Data: 12 August 2018 17:44:50]
[Treno richiedente autorizzazione: T3], [Segmento attuale: MA16], [Segmento richiesto: MA12], [Autorizzato: SI], [Data: 12 August 2018 17:44:53]
[Treno richiedente autorizzazione: T2], [Segmento attuale: MA11], [Segmento richiesto: MA12], [Autorizzato: NO], [Data: 12 August 2018 17:44:53]
[Treno richiedente autorizzazione: T4], [Segmento attuale: MA2], [Segmento richiesto: MA1], [Autorizzato: SI], [Data: 12 August 2018 17:44:53]
[Treno richiedente autorizzazione: T5], [Segmento attuale: MA3], [Segmento richiesto: MA2], [Autorizzato: SI], [Data: 12 August 2018 17:44:53]
[Treno richiedente autorizzazione: T1], [Segmento attuale: MA7], [Segmento richiesto: MA3], [Autorizzato: SI], [Data: 12 August 2018 17:44:53]
[Treno richiedente autorizzazione: T3], [Segmento attuale: MA12], [Segmento richiesto: S8], [Autorizzato: SI], [Data: 12 August 2018 17:44:56]
[Treno richiedente autorizzazione: T2], [Segmento attuale: MA11], [Segmento richiesto: MA12], [Autorizzato: SI], [Data: 12 August 2018 17:44:56]
[Treno richiedente autorizzazione: T4], [Segmento attuale: MA1], [Segmento richiesto: S1], [Autorizzato: SI], [Data: 12 August 2018 17:44:56]
[Treno richiedente autorizzazione: T1], [Segmento attuale: MA3], [Segmento richiesto: MA8], [Autorizzato: SI], [Data: 12 August 2018 17:44:56]
[Treno richiedente autorizzazione: T5], [Segmento attuale: MA2], [Segmento richiesto: MA1], [Autorizzato: SI], [Data: 12 August 2018 17:44:56]
[Treno richiedente autorizzazione: T1], [Segmento attuale: MA8], [Segmento richiesto: S6], [Autorizzato: SI], [Data: 12 August 2018 17:44:59]
[Treno richiedente autorizzazione: T5], [Segmento attuale: MA1], [Segmento richiesto: S1], [Autorizzato: SI], [Data: 12 August 2018 17:44:59]
[Treno richiedente autorizzazione: T2], [Segmento attuale: MA12], [Segmento richiesto: S8], [Autorizzato: SI], [Data: 12 August 2018 17:44:59]
```

- T1.log

```
[Attuale: S2], [Prossimo: MA5], 12 August 2018 17:44:43
[Attuale: MA5], [Prossimo: MA6], 12 August 2018 17:44:47
[Attuale: MA6], [Prossimo: MA7], 12 August 2018 17:44:50
[Attuale: MA7], [Prossimo: MA3], 12 August 2018 17:44:53
[Attuale: MA3], [Prossimo: MA8], 12 August 2018 17:44:56
[Attuale: MA8], [Prossimo: S6], 12 August 2018 17:44:59
[Attuale: S6], [Prossimo: ---], 12 August 2018 17:45:02
```

- T2.log

```
[Attuale: S3], [Prossimo: MA9], 12 August 2018 17:44:43
[Attuale: MA9], [Prossimo: MA10], 12 August 2018 17:44:47
[Attuale: MA10], [Prossimo: MA11], 12 August 2018 17:44:50
[Attuale: MA11], [Prossimo: MA12], 12 August 2018 17:44:53
[Attuale: MA11], [Prossimo: MA12], 12 August 2018 17:44:56
[Attuale: MA12], [Prossimo: S8], 12 August 2018 17:44:59
[Attuale: S8], [Prossimo: ---], 12 August 2018 17:45:02
```

- T3.log

```
[Attuale: S4], [Prossimo: MA14], 12 August 2018 17:44:43
[Attuale: MA14], [Prossimo: MA15], 12 August 2018 17:44:47
[Attuale: MA15], [Prossimo: MA16], 12 August 2018 17:44:50
[Attuale: MA16], [Prossimo: MA12], 12 August 2018 17:44:53
[Attuale: MA12], [Prossimo: S8], 12 August 2018 17:44:56
[Attuale: S8], [Prossimo: ---], 12 August 2018 17:44:59
```

- T4.log

```
1  [Attuale: S6], [Prossimo: MA8], 12 August 2018 17:44:43
2  [Attuale: MA8], [Prossimo: MA3], 12 August 2018 17:44:47
3  [Attuale: MA3], [Prossimo: MA2], 12 August 2018 17:44:50
4  [Attuale: MA2], [Prossimo: MA1], 12 August 2018 17:44:53
5  [Attuale: MA1], [Prossimo: S1], 12 August 2018 17:44:56
6  [Attuale: S1], [Prossimo: ---], 12 August 2018 17:44:59
```

- T5.log

```
1  [Attuale: S5], [Prossimo: MA4], 12 August 2018 17:44:43
2  [Attuale: MA4], [Prossimo: MA3], 12 August 2018 17:44:47
3  [Attuale: MA4], [Prossimo: MA3], 12 August 2018 17:44:50
4  [Attuale: MA3], [Prossimo: MA2], 12 August 2018 17:44:53
5  [Attuale: MA2], [Prossimo: MA1], 12 August 2018 17:44:56
6  [Attuale: MA1], [Prossimo: S1], 12 August 2018 17:44:59
7  [Attuale: S1], [Prossimo: ---], 12 August 2018 17:45:02
```

Rbc ha negato solo due volte l'autorizzazione per muoversi ad un treno. In questo caso i treni bloccati sono stati T2 e T5. E ciò non è una casualità data dalla scheduler, ma una conseguenza dell'ottima sincronia tra processi. Indipendentemente dal numero di volte in cui verrà lanciato il programma, a prescindere dalla modalità, i treni che dovranno rimanere fermi un turno saranno sempre e solo due. Ovviamente ciò vale per i particolari percorsi presi in esame, ma in generale viene garantito che ad ogni ciclo tutti i treni che si possono muovere lo faranno.

Compilazione ed avvio

La compilazione dell'intero progetto può essere effettuata sfruttando il comando *make* (o *make all*). Gli eseguibili e i file oggetto verranno messi nella cartella *bin*. È possibile inoltre compilare singolarmente i tre eseguibili con *make ertms*, *make train* e *make rbc*. Per avviare il progetto si dovrà lanciare *ertms* con i parametri opportunamente scelti (non viene fatta distinzione tra maiuscole e minuscole). Non è necessaria l'uso di una working directory particolare, dal momento che il programma riesce a ricavare la sua posizione assoluta. Bisogna tuttavia preservare i percorsi relativi tra gli eseguibili e la cartella *resources*. Si consiglia inoltre di far terminare spontaneamente il programma, per permettere che tutte le risorse usate vengano correttamente rilasciate. La compilazione ed esecuzione del progetto sono state testate sulla versione 18.04 di Xubuntu a 32 e 64 bit.

Documentazione completa delle funzioni

Common

- **buildPathMAxFile**
  Combina le varie macro per costruire il path del file *MA* richiesto. Restituisce un puntatore all'inizio della stringa.
- **buildPathRouteFile**
  Combina le varie macro per costruire il path del file route richiesto. Restituisce un puntatore all'inizio della stringa.
- **buildPathTrainLogFile**
  Combina le varie macro per costruire il path del file log del treno richiesto. Restituisce un puntatore all'inizio della stringa.
- **buildPathRbcLogFile**
  Combina le varie macro per costruire il path del file log di *rbc*. Restituisce un puntatore all'inizio della stringa.
- **buildPathTrainSocketFile**
  Combina le varie macro per costruire il path del socket per la comunicazione con i treni. Restituisce un puntatore all'inizio della stringa.
- **buildPathErtmsSocketFile**
  Combina le varie macro per costruire il path del socket per la comunicazione con *ertms*. Restituisce un puntatore all'inizio della stringa.
- **csprintf**
  Formatta i parametri ricevuti in modo analogo alla funzione *sprintf*, ma al contrario di questa alloca autonomamente la stringa dove depositare il risultato della formattazione. Restituisce un puntatore all'inizio della stringa.
- **setUpExeDirPath**
  Riceve il percorso assoluto dell'eseguibile e lo copia all'interno di una variabile globale. Dopodiché tronca il nome dell'eseguibile dalla stringa.
- **truncExeName**
  Inserisce un carattere terminatore nella stringa ricevuta dopo l'ultimo carattere '/'. Restituisce il puntatore della stringa.
- **createDirIfNotExist**
  Crea la cartella specificata in caso non esistesse.
- **waitChildrenTermination**
  Attende la terminazione di un numero di processi figli pari a quanto ricevuto tramite parametro.

- **setUpSocket**
  Costruisce un socket anonimo. Tramite un booleano tale socket viene legato al nome specificato oppure tentato di connettere a un socket già esistente.

Ertms

- **main**
  Funzione di avvio del processo *ertms*. Azzera un'eventuale *umask*, calcola la propria posizione assoluta e, in base ai parametri di lancio, configura e avvia i processi train o il processo *rbc*.
- **getExePath**
  Ricava la posizione assoluta del processo corrente, registrandola in un'apposita stringa. Restituisce un puntatore all'inizio della stringa.
- **launchEtsc**
  Prepara le variabili condivise che verranno usate dai treni. Dopodiché lancia i processi i treni attendendone la terminazione. Alla fine dealloca le variabili condivise.
- **setUpSharedVariableForTrains**
  Alloca nell'area di memoria condivisa una struttura contenente variabili che verranno usate dai treni. Inoltre inizializza il mutex e la condizione variabile.
- **cleanUpSharedVariableForTrains**
  Pulisce le variabili condivise usate dai treni: distrugge il mutex e la condizione variabile, rilasciando l'area di memoria condivisa.
- **launchRbc**
  Prepara le variabili condivise che verranno usate da *rbc*. Dopodiché lancia il processo *rbc*, gli passa i vari percorsi dei treni e ne attende la terminazione. Alla fine dealloca le variabili condivise.
- **setUpSharedVariableForRbc**
  Alloca nell'area di memoria condivisa una struttura contenente variabili che verranno usate da *rbc*. Inoltre inizializza tutti i vari mutex e gli array di interi presenti.
- **initializeIntArray**
  Imposta tutti gli elementi dell'array ricevuto a 0.
- **cleanUpSharedVariableForRbc**
  Pulisce le variabili condivise usate da *rbc*: distrugge tutti i mutex, rilasciando l'area di memoria condivisa.
- **createMAxFiles**
  Crea o tronca tutti i file *MA* necessari al funzionamento dei treni.
- **startTrains**
  Avvia tutti i processi treni necessari, passandogli la loro posizione assoluta, un numero identificativo e la modalità di avvio.

- **startRbc**
  Avvia il processo *rbc*, passandogli la sua posizione assoluta.
- **sendRoutes**
  Si connette al socket creato dal processo *rbc* e gli invia tutti i percorsi dei treni letti dagli appositi file.

Log

- **logTrain**
  Formatta e stampa sull'apposito file di log il messaggio di log richiesto dal treno.
- **logRbc**
  Formatta e stampa sull'apposito file di log il messaggio di log richiesto da rbc.
- **getLogMessage**
  Combina i vari parametri ricevuti e, usando un booleano, decide quale template usare per la formattazione. Restituisce un puntatore all'inizio della stringa.
- **formatTime**
  Formatta la data e l'ora corrente depositando il risultato in una variabile globale.
- **logOnFile**
  Scrive il messaggio ricevuto sul file richiesto.

Rbc

- **main**
  Funzione di avvio del processo *rbc*. Imposta le variabili condivise, riceve i percorsi dei treni e si mette in attesa di connessioni in ingresso. Alla fine dealloca le variabili condivise.
- **setUpSharedVariable**
  Crea un puntatore all'area di memoria condivisa creata da *ertms*.
- **cleanUp**
  Dealloca tutte le variabli globali e rilascia l'area di memoria condivisa.
- **importRoutes**
  Crea un socket e attende la connessione di *ertms*. Legge quindi i vari percorsi dei treni e li immagazzina in una particolare struttura a lista. Imposta inoltre lo stato delle stazioni in base al numero di treni presenti.
- **startTrainSocket**
  Crea un socket e attende la connessione dei treni, generando un processo figlio per ognuno di essi. Attende la terminazione dei processi generati.

- **serveTrain**
  In un ciclo infinito attende la notifica della posizione di un treno e la sua successiva richiesta di spostamento. Al termine del ciclo rilascia eventuali mutex posseduti.
- **waitForPosition**
  Riceve la posizione di un treno lockando il mutex corrispondente. Risponde al treno con un messaggio affermativo. Restituisce -1 se il treno ha chiuso la connessione, 0 altrimenti.
- **waitForRequest**
  Riceve la richiesta di movimento di un treno lockando il mutex corrispondente al segmento desiderato. Controlla se tale movimento è possibile: in caso positivo aggiorna lo stato dei segmenti e risponde affermativamente, viceversa risponde negativamente.
- **updatePosition**
  Fa scorrere la posizione attuale del treno e il suo indice. A seconda del movimento effettuato aggiorna lo stato delle stazioni e dei segmenti.
- **unlockMutex**
  Rilascia il mutex specificato.

Route

- **generateRoute**
  Decodifica la stringa rappresentante il percorso di un treno in una struttura a lista in cui ogni nodo punta al successivo. Restituisce il puntatore al primo nodo.
- **readLine**
  Legge dal descrittore richiesto fino al primo carattere di ritorno a capo (‘\n’) o il primo carattere terminatore (‘\0’), scartando eventuali caratteri spazio. Restituisce un puntatore all’inizio della stringa letta.
- **buildNode**
  Costruisce il nodo corrispondente alla stringa specificata. Il nodo conterrà un numero positivo per i segmenti e uno negativo per le stazioni. Restituisce il puntatore al nodo costruito.
- **decodeId**
  Converte il codice di un nodo nella sua rappresentazione a stringa. Restituisce un puntatore all’inizio della stringa.
- **destroyRoute**
  Dealloca tutti i nodi di un percorso.

Train

- **main**
  Funzione di avvio dei processi treni. Imposta le variabili condivise e i vari puntatori a funzione, legge e decodifica il proprio percorso e infine avvia il movimento. Alla fine dealloca le variabili condivise.
- **setUpSharedVariable**
  Crea un puntatore all'area di memoria condivisa creata da *ertms*.
- **cleanUp**
  Dealloca tutte le variabli globali, rilascia l'area di memoria condivisa e chiude il socket.
- **connectToSocket**
  Si connette al socket creato da *rbc* e deposito il relativo descrittore in un'apposita variabile globale.
- **readAndDecodeRoute**
  Legge il proprio percorso dall'apposito file e lo trasforma in una struttura a lista. Restituisce il puntatore al primo nodo.
- **requestModeEtcs1**
  Controlla se il segmento richiesto è libero leggendo i file *MA*. Restituisce 1 se il movimento è autorizzato, 0 altrimenti.
- **checkMAxFile**
  Locka il file *MA* richiesto e legge il primo carattere. Restituisce 1 se viene letto uno 0, 0 altrimenti.
- **requestModeEtcs2**
  Manda un messaggio a *rbc* chiedendo l'autorizzazione a muoversi. In caso di risposta affermativa apre il file *MA* corrispondente al segmento richiesto e restituisce 1, 0 altrimenti.
- **startTravel**
  Cicla finché il treno non è arrivato a destinazione. Attende che tutti i treni siano pronti, notifica la propria posizione secondo la modalità opportuna, attende nuovamente che siano tutti pronti, logga la sua posizione e richiede l'autorizzazione a spostare secondo la modalità opportuna. In caso di risposta affermativa il treno si sposta e aggiorna la propria posizione. Prima di sospendersi in attesa del nuovo ciclo vengono chiusi tutti i file aperti.
- **waitOtherTrains**
  Locka un mutex apposito e controlla che tutti i treni siano pronti. Alla fine sblocca il mutex.

- **checkOtherTrains**
  Incrementa la variabile opportuna e calcola la somma tra i treni in attesa e quelli che hanno completato il proprio viaggio. Se tale somma è pari al numero di treni in esecuzione vengono svegliati tutti i treni in attesa, altrimenti se il treno non ha ancora completato il suo giro si sospende inserendosi nella lista dei treni in attesa.
- **eLUltimoChiudaLaPorta**
  Vengono risvegliati tutti i treni in attesa, azzerando il loro contatore.
- **travelCompleted**
  Locka un mutex apposito e controlla che tutti i treni siano pronti. Alla fine sblocca il mutex.
- **lockExclusiveMA**
  Apre il file *MA* richiesto e vi inserisce un lock esclusivo.
- **openFile**
  Apre il file *MA* richiesto e deposita il descrittore corrispondente nella variabile specificata. Restituisce 1 se l'apertura ha avuto successo, 0 altrimenti.
- **closeFile**
  Chiude il file specificato, rilasciando eventuali lock.
- **notifyPosition**
  Manda un messaggio a *rbc* segnalando l'attuale posizione. Apre il file corrispondente e attende la risposta di *rbc*.
- **move**
  Scrive 0 nel file *MA* corrispondente all'attuale posizione e 1 nel file *MA* corrispondente alla prossima tappa del treno.
- **writeOneByte**
  Scrive la stringa specificata all'inizio del file desiderato, assicurandosi che tali modifiche raggiungano il disco.