UNIVERSITÀ
DEGLI STUDI
FIRENZE

Operating Systems Laboratory Project

Ali Al-azzawi

Content

1- The purpose of the project :

The purpose of the project is to simulate the behavior of a set of trains that cross multiple track segments. The railway mission of each train is to reach a specific station. The main constraint on train movement is that each track segment can be occupied by only one train at a time. Each train receives permission to access the next track segment during its railway mission. The project implements the above schema in two different ways.

2- The tools:

The tools used to implement the project are Ubuntu OS installed on virtualbox and C language.

3- How I organized my code :

3-1- main.c :

The main process from there, if not RBC, I launch the train_father process with MODE, MAP arguments, else I launch the RBC process (rbc.c) with MAP argument. I proceed with fork() then execl() to launch these processes.

3-2- train_process.c :

Here first I create a log/Tx.log file then separate functions depending on the mode :

A- If mode is ETCS1 : I first retrieve an itinerary string in the register/MAPx.txt file, saving only the line relevant to the train name (Tx) and skipping the other lines, so now I have an itinerary string.

B- If mode is ETCS2 : first I connect up to the RBC server, then I send a message, « ASKMAP <map> <train_name> », to the server. It sends back an itinerary string.

For both modes, then I transform this string into a linked list. The current stop is the list's head (its first node), the next stop is the list's second node. A train has arrived to destination if there is no second node. Now begins the main loop for each train process, that runs while the train hasn't arrived to its destination :

If mode is ETCS1 : first I wait <train_number>*2 seconds, so that the timing is different for each train_process while reading the MAx segment files. I check the next stop. If the stop is not a segment, we can go to it, else if the file corresponding to that stop is 0 so the train can go to it. If the train can go to the next stop, then it check if the current stop is a segment. If yes, put 0 inside. Then check if the next stop is a segment. If yes, put 1 inside. Loop back after 2 seconds of waiting time.

If mode is ETCS2 : I should still be connected to the RBC server. The train process sends a message to RBC : « REQ <train_name> <current_stop> <next_stop> ». If the RBC server answers « 0 », the train

across to the next segment else the answer «1» so wait in the same segment. Loop back after 2 seconds. End the server connection if not looping back.

3-3- train_father.c :

from there, I create 16 MAx.txt files in the Max folder, all initialized with 0 inside. then fork() 5 times to concurrently launch 5 train_processes with execl(). I give as argument the MODE, MAP and their train name (T1 to T5) to the train_processes.

3-4- train_process_utils.c :

This file is linked list implementation, used in train_process.c.

3-5- rbc.c :

I first create the socket server. Then wait for 5 connections (the train processes). then begin a loop that continuously reads incomming data from the train processes, runs as long as receive data from them. if I receive « ASKMAP <map> <train_name> », then I retrieve the corresponding itinerary string in the same manner, in the register/MAPx.txt files. I then send it back to the train process.

If I receive « REQ <train_name> <current_stop> <next_stop> », then check to see if the segment is free from the MAx files in the same manner as the train_process does when launched in ETCS1 mode. If next segment is free, then mark it as busy (1) and mark the old one as free (0), of course only if they are segments, otherwise if they are stations I do nothing. Whenever stop receiving data, the RBC server is stopped.

3-6- *.h :

header files, function prototypes for each file.

4- Execution:

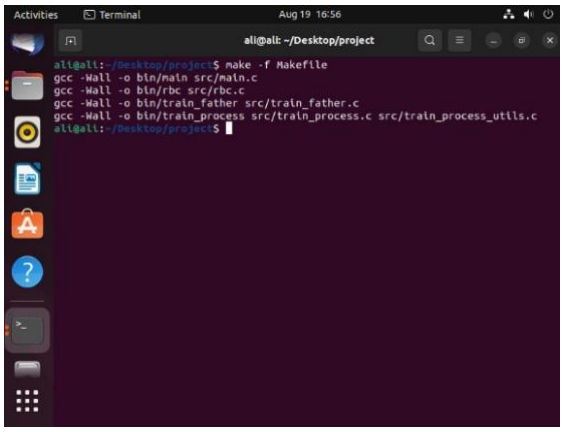
Run «make » in the project directory to compile the code.

make -f Makefile

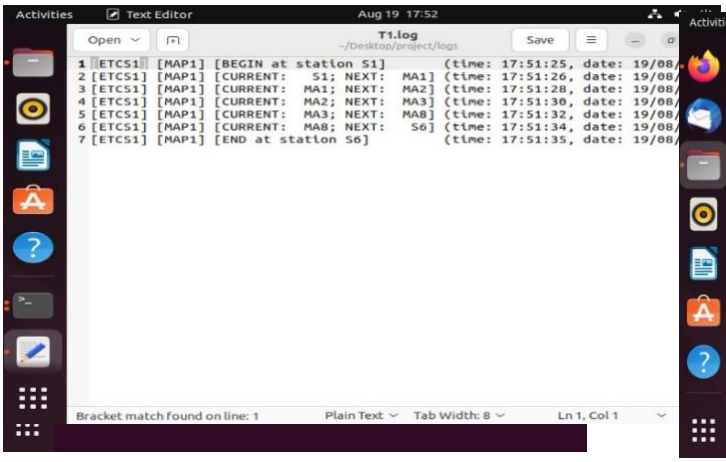
Then to choose one of those modes:

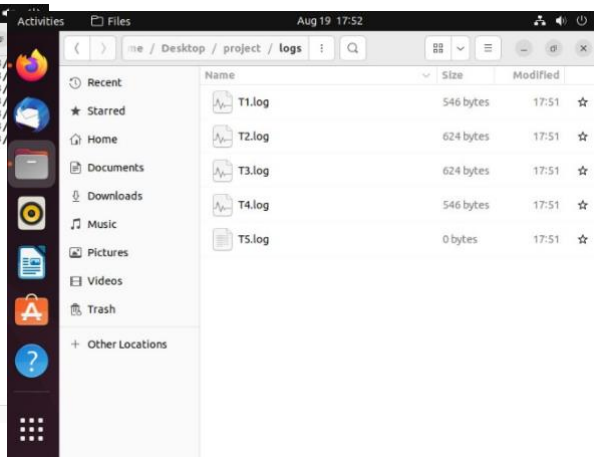
A- ./bin/main ETCS1 MAP1
B- ./bin/main ETCS1 MAP2
C- ./bin/main ETCS2 MAP1 and ./bin/main ETCS2 RBC MAP1 in the same time but in two shells.
D- ./bin/main ETCS2 MAP2 and ./bin/main ETCS2 RBC MAP2 in the same time but in two shells.

5- The results after execution for each mode:

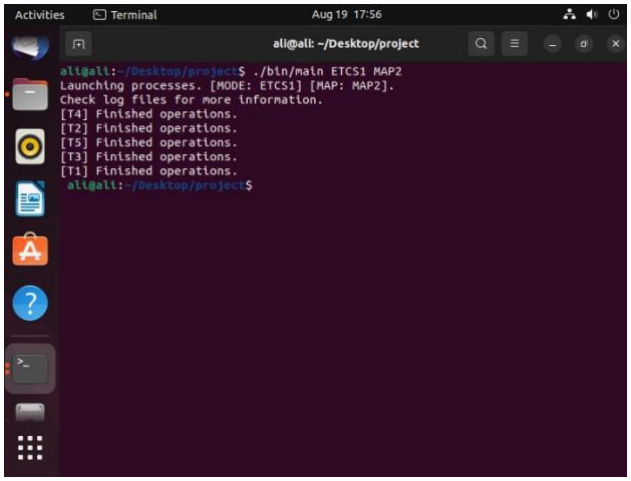


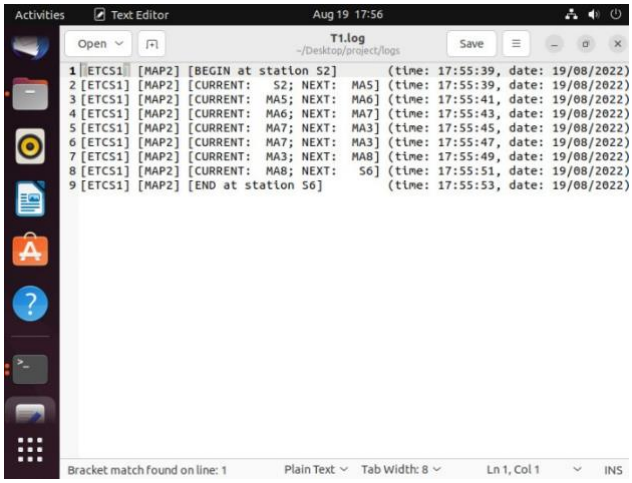
5-1- ./bin/main ETCS1 MAP1



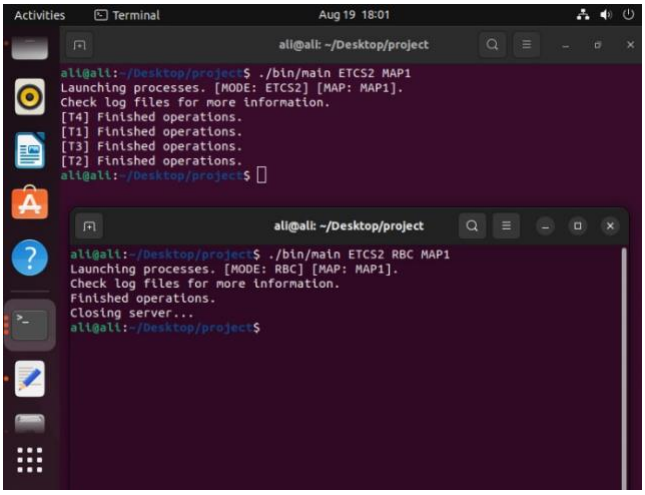


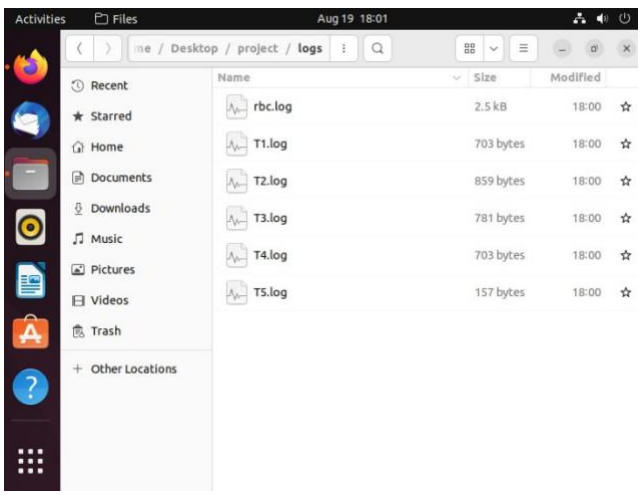
5-2- ./bin/main ETCS1 MAP2

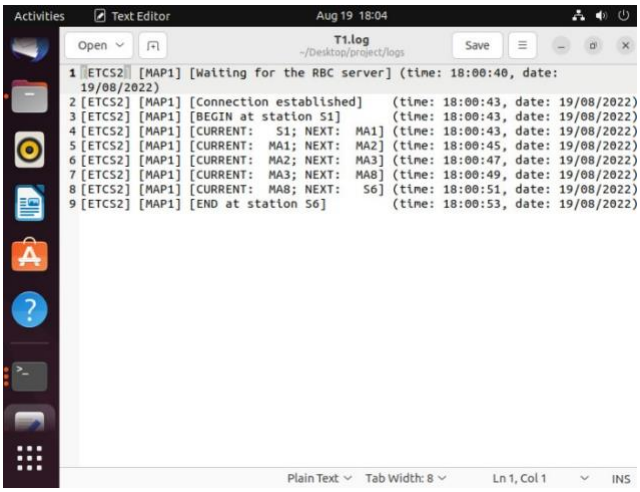




5-3-. In/main ETCS2 MAP1 and ./bin/main ETCS2 RBC MAP1







5-4- in/main ETCS2 MAP2 and ./bin/main ETCS2 RBC MAP2