



Compte rendu STA

Gestion de trains par RBC

HUO Jiaxi / ZHANG Chongmo

13/10/2019

Table des matières

1.	INTRODUCTION :	3
2.	ANALYSE	5
3.	SCRIPTS :	6
4.	EXECUTION DES PROGRAMMES.	14
	LA MISE À JOUR DE LA POSITION DU TRAIN	16
	LA MISE À JOUR DE LA FIN D'AUTORISATION DE MOUVEMENT D'UN TRAIN.	17
	LA SUPPRESSION D'UN TRAIN D'UN RBC	18
	LA LECTURE DE LA FIN D'AUTORISATION DE MOUVEMENT D'UN TRAIN.	19
	L'ENQUETE D'AUTORISATION D'UN NOUVEAU TRAIN	19
	QUITTER L'APPLICATION	20
5.	BILAN :	21

1. Introduction :

Contrôle et signalisation ferroviaire :

Le contrôle et signalisation ferroviaire est réalisé par la communication entre RBC (Radio Bloc Center) et calculateur EVC (European Vital Computer). Ces applications fonctionnent afin d'assurer la sécurité de trains.

Dans ce travail, nous avons utilisé Socket en C (Prototype de réseaux offert par langage C) pour simuler la communication entre RBC et EVC. Nous créons les sockets sur le serveur (RBC) et le client (EVC). Et les sockets permettent l'échange de données entre RBC et EVC.

Les deux programmes du réseau échangent des données via une connexion de communication bidirectionnelle, dont l'une des extrémités est appelée socket. Socket est la couche d'abstraction logicielle intermédiaire que la couche application communique avec la famille de protocoles TCP / IP, à savoir un ensemble d'interfaces.

Prototype de fonction socket() :

```
1. int socket(int domain, int type, int protocol);
```

domain: Le domaine de protocole, également appelé famille de protocoles. Les familles de protocoles couramment utilisées sont AF_INET, AF_INET6, AF_LOCAL (ou AF_UNIX, Socket de domaine Unix), AF_ROUTE, etc. La famille de protocoles détermine le type d'adresse de la socket. L'adresse correspondante doit être utilisée dans la communication, par exemple AF_INET détermine la combinaison de l'adresse ipv4 (32 bits) et du numéro de port (16 bits). AF_UNIX détermine qu'un chemin absolu doit être utilisé. Nom comme adresse. Dans notre travail, nous utilisons AF_INET

type : Le type de protocole, spécifie le type de socket. Les types de socket les plus couramment utilisés sont SOCK_STREAM, SOCK_DGRAM, SOCK_RAW, SOCK_PACKET, SOCK_SEQPACKET, etc. Le Streaming Socket (SOCK_STREAM) est un Socket orienté connexion pour les applications de service TCP orienté connexion. Datagram Socket (SOCK_DGRAM) est un socket sans connexion qui correspond à une application de service UDP sans connexion.

Le service UDP offre la simple structure et il satisfait notre travail. UDP permet le transport d'un simple paquet de données, vu comme un tout, à destination d'un port bien défini.¹ Donc nous appliquons UDP sur notre travail.

Protocol : Le prototype spécifie l'accord. Les protocoles courants incluent IPPROTO_TCP, IPPROTO_UDP, IPPROTO_STCP, IPPROTO_TIPC, etc., correspondant au protocole de transport TCP, au protocole de transport UDP, au protocole de transport STCP et au protocole de transport TIPC. **Protocol = 0 : Le protocole par défaut correspondant au deuxième type de paramètre est automatiquement sélectionné.**

Valeur de retour

Si l'appel aboutit, il renvoie le descripteur du socket nouvellement créé, sinon INVALID_SOCKET (**impossible de renvoyer -1 sous Linux**).

Un descripteur de socket est une valeur de type entier. Il existe une table de descripteur de socket dans l'espace de processus de chaque processus, qui stocke la correspondance entre le descripteur de socket et la structure de données de socket. La table a un champ pour le descripteur du socket nouvellement créé et un autre champ pour l'adresse de la structure de données du socket, afin que la structure de données du socket correspondante puisse être trouvée en fonction du descripteur de socket. Chaque processus a une table de descripteur de socket dans son propre espace de processus, mais la structure de données du socket se trouve dans la mémoire tampon du noyau du système d'exploitation.

Prototype de fonction bind() :

```
1. int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
```

La fonction bind() attribue une adresse spécifique dans une famille d'adresses au socket. Par exemple, AF_INET et AF_INET6 attribuent une combinaison adresse IPv4 ou ipv6 et numéro de port à la prise.

Socketfd : Socket qui identifie la prise connectée.

address : Un pointeur de structure sockaddr contenant l'adresse et le numéro de port à combiner.

address_len : Déterminer la taille du tampon d'adresse.

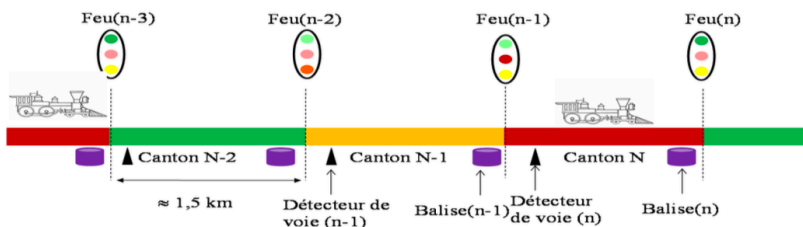
¹ http://www.fil.univ-lille1.fr/~noe/rsx_portail/files/Slide6_Transport.pdf

Valeur de retour

Si la fonction s'exécute correctement, la valeur renvoyée est 0, sinon c'est `SOCKET_ERROR`.

2. Analyse

Afin de réaliser la gestion de trains par RBC, nous devrions créer une application, celle qui peut enregistrer les informations de trains sur la ligne au sol, les trains peuvent mettre à jour leurs informations et faire l'enquête des autorisations de mouvement.



Chaque train s'occupe un canton.

L'autorisation de mouvement signifie que l'autorisation d'un train qui va entrer certain canton. Comme indiqué dans la figure ci-dessus, le canton N-1 est la fin d'autorisation de mouvement (EOA), en raison d'occupation de canton N, le train n'a pas d'autorisation d'entrer le canton N. Canton N-1 est la fin d'autorisation de mouvement.

Chaque train a sa propre fin d'autorisation de mouvement et sa localisation. Donc notre système permettra d'enregistrer les fins d'autorisation de mouvement et les localisations de tous les trains sur la ligne. Le train peut faire l'enquête au système par fourni sa localisation. Le système va faire la comparaison entre la localisation d'un train qui fait l'enquête et les localisations des train qui existent dans la ligne.

Toutefois, lorsque le train entre dans le canton N-1, il doit décélérer et ne peut pas passer à la vitesse initiale. Par conséquent, lorsque le train pénètre dans le canton N-1, il reçoit la réponse de RBC « Passage Interdit » et le train décélère.

Le train ne peut pas être dans le même canton que les autres trains.

Les travaux à réaliser :

- a. L'enregistrement d'un train dans le RBC,
- b. La suppression d'un train d'un RBC,
- c. La mise à jour de la position du train,
- d. La lecture de la fin d'autorisation de mouvement d'un train,
- e. La mise à jour de la fin d'autorisation de mouvement d'un train.
- f. L'enquête d'autorisation d'un nouveau train.

3. Scripts :

serveur_train_enreg_fin.c : RBC d'enregistrer les information de trains.

client_train_enreg_fin.c : EVC d'envoyer l'information de train à RBC pour être enregistré.

serveur_train_rev.c : RBC d'envoyer l'autorisation de mouvement à un train qui émit l'enquête.

client_train_rev.c : EVC d'envoyer l'enquête à RBC pour obtenir l'autorisation de mouvement.

sup_train_fin : La suppression d'un train d'un RBC.

mis_a_jour_local_fin.c : La mise à jour de la position du train.

Lecture_train_EOA.c : La lecture de la fin d'autorisation de mouvement d'un train.

mis_a_jour_eoa_fin.c : La mise à jour de la fin d'autorisation de mouvement d'un train.

Test_general.c : Application qui permette de mettre en œuvre le contrôle réalisé par un RBC.

Chaque script correspond à un programme à exécuter, celui qui contient au dossier.

4. Analyse des scripts :

La communication entre RBC et EVC est réalisée par la liaison de sockets.

Pour RBC :

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <sys/types.h>
4. #include <sys/socket.h>
5. #include <netinet/in.h>
6. #include <netdb.h>
7. #include <arpa/inet.h>
8. #include <string.h>
9. #include <unistd.h>
10.
11. //Definition du port du serveur (Le client va connecter au serveur par ce port)
```

```

12. #define portserveur 3306 // Faut correspondre au port de client(On peut choisir le port aléatoirement)
13. //Definition de la taille maximale des chaines de caracteres
14. #define MAXCAR 80 // Faut dépasser 6.
15. #define N 100 //Le nombre de train sur la ligne
16.
17. int main()
18. {
19. printf("Appuyez sur la touche Entrée pour commencer\n");
20. getchar();// Ce méthode est pour eviter les caractères de fichier MEMO tronqués apparaissent après la réexécution de ce script.
21. int sd; /* Socket de dialogue */
22. int identifiant; // identifiant du train
23. int posi;// position du train
24. int eoa;//EOA de train
25. FILE *fp; // Le pointer de fichier qui enregistre l'information de train
26. int i;
27. p=tr;
28. p1=tr1;
29.
30. fclose(fp);
31.
32. struct sockaddr_in adrecliv ; // Adresse du client
33.
34. struct sockaddr_in adrlect ; // Adresse du serveur
35.
36. char bufflect[MAXCAR]; // Buffer en reception
37. char buff[MAXCAR]; // Buffer en emission
38. char ges[MAXCAR];
39.
40. int infoecu; //Permet d'enregistrer le nombre d'octets recus
41. int infoemis; //Permet d'enregistrer le nombre d'octes emis
42.
43. int taille; //Permet de recuperer la taille en octets de l'adresse d'un client
44. int status;
45. /*code status renvoie par une fonction. Si vaut SOCKET_ERROR c'est qu'il y a une erreur du système */
46.
47. // Creation de la socket qui correspond a un point de dialogue
48. sd=socket(AF_INET, SOCK_DGRAM, 0);
49. /* Creation de socket,
50. AF_INET détermine la combinaison de l'adresse ipv4 (32 bits) et du numéro de port (16 bits).
51. SOCK_DGRAM détermine l'utilisation de service UDP
52. Protocol = 0 : Le protocole par défaut correspondant au deuxième type de paramètre est automatiquement sélectionné. Dans ce cas, prototype = IPPROTO_UDP
53. */
54. if(sd < 0)
55. {
56.     printf("Il existe l'erreur de creation d'une socket \n");
57.     exit(-1);
58. }
59.
60. //Definition de l'adresse du serveur
61. adrlect.sin_family=AF_INET;
62. adrlect.sin_port=htons(portserveur); //Affectation du port du serveur. On pourra le voir au niveau system par la commande "netstat -an"
63. adrlect.sin_addr.s_addr=INADDR_ANY; // Indique que le serveur recoit les data envoye a n'importe laquelle de ses adresses IP
64.
65.
66. // Associer cette adresse a la socket du lecteur pour qu'il puisse recevoir des data
67.
68. status=bind(sd, (const struct sockaddr *) &adrlect, sizeof(adrlect));
69.
70. if(status != 0)
71. {
72.     printf("Il existe l'erreur de creation d'une socket \n");
73.     exit(-1);
74. }
75.
76. printf("\n Attente la reponse du serveur \n\n");

```

Pour EVC :

Les identifiants sont des chaînes de 6 caractères. Pour distinguer chaque train, nous assignons un numéro chiffré pour chaque train. C'est-à-dire que chaque train porte un identifiant (Par exemple TGV134), un numéro chiffré (1, 2, 3 ...), sa localisation et sa EOA.

```
1. struct train{
2.     int num; //le numéro de train (1, 2, 3...)
3.     char ID[MAXCAR]; // l'identifiant de train
4.     int position; // Position de train
5.     int End; // EOA de train
6. }tr[N],*p,tr1[N],*p1;
```

Le numéro d'un train est géré automatiquement. (Le premier train porte 1, le deuxième train porte 2 ...)

Pour enregistrer les informations des trains sur la ligne, nous créons un fichier.txt afin de sauvegarder les informations. Il est facile à faire les opérations sur les informations des trains en utilisant le fichier.

```
1. if( (fp=fopen("MEMO.txt", "w+")) == NULL ){
2.     printf("Erreur sur l'ouverture de fichier, veuillez vous sortir!\n");
3.     getchar();
4.     exit(1);
5. }
6. fread(tr, sizeof(struct train), N, fp);
7.
8. p=p+(j-1);
9. p->num = j;
10. strcpy(p->ID, ges);
11. p->position=posi;
12. p->End=100;
13.
14. /* Sauvegarder les informations des trains en utilisant le pointer d'un fichier */
15.
16. fwrite(tr, sizeof(struct train), N, fp);
17. p=p-(j-1);
18. rewind(fp);
19.
20.
21. fread(tr1, sizeof(struct train), N, fp);
22. printf("\nN° de train Localisation\n");
23. for(i=0; i<j; i++,p1++){
24.     printf("%s\t\t%d\n", p1->ID, p1->position);
25. }
26. printf("\n");
27. p1=p1-j; //Retourner au premier
28. j++;
29. fclose(fp);
```

Après avoir reçu les informations de localisation de tous les trains, le système RBC attribue automatiquement un EOA à tous les trains et la EOA = 100 du train le plus proche de la gare.

```
1. int m;
2. int min = 100;
```



```

3.
4.
5.
6. fread(tr, sizeof(struct train), N, fp);
7. fread(tr1, sizeof(struct train), N, fp);
8. for(i=0; i<N; i++,p1++)
9. {
10.     min = 100;
11.     //min = p->position - p1->position;
12.     for(m=0; m<N; m++,p++)
13.     {
14.         if(min > (p->position - p1->position) && (p->position - p1->position) > 0)
15.         {
16.             min = p->position - p1->position;
17.             p1->End = p->position - 1;
18.         }
19.     }
20.     p = p - N;
21.     //p1->End=min - 1;
22. }
23.
24. p1=p1-N;
25. rewind(fp);
26. fclose(fp);
27.
28. fpn = fopen("MEMO.txt", "wb"); // Vider le fichier initial
29. fclose(fpn);
30.
31. fpn = fopen("MEMO.txt", "w+");
32. printf("\nN° de train Localisation EOA\n") ;
33.
34. for(i=0; i<N; i++,p1++){
35.     if(p1->num !=0)
36.     printf("%s\t\t%d\t\t%d\n", p1->ID, p1->position, p1->End);
37.     fwrite(tr1, sizeof(struct train), N, fpn);
38. } // Remplir le fichier initial
39.
40.     p1=p1-N;
41.     rewind(fpn);
42.     fclose(fpn);

```

Pour traiter les informations de trains, nous faisons l'opération sur le fichier MEMO.txt. Par exemple, afin de supprimer un train sur la ligne, il faut justement supprimer une ligne appartenant à ce train dans le fichier :

```

1. FILE* fp = fopen("MEMO.txt", "r");
2. FILE* fpn;
3. //FILE* fpt = fopen("temp.txt", "r");
4. //p=tr;
5. p1=tr1;
6. p=tr;
7.
8. //FILE* fpt = fopen("temp.txt", "w");
9. int i = 0;
10. rewind(fp);
11. fread(tr1, sizeof(struct train), N, fp);
12. //printf("\nN° de train Localisation\n") ;
13. for(i=0; i<N; i++,p1++){
14.
15.     if(strcmp(p1->ID ,buf)==0)
16.     {
17.         flag = 1;
18.         continue; // Supprimer les informations du train qui est supprimé
19.     }
20.

```

```

21.     p->num = p1->num;
22.     strcpy(p->ID,p1->ID);
23.     p->position=p1->position;
24.     p->End=p1->End;
25.     p=p+1; // Sauvegarder les trains restants
26.
27. }
28. if(flag == 0)
29. {
30.     printf("Je ne trouve pas votre train désolé. :( \n");
31.     exit(-1);
32. }
33. printf("\n");
34. p1=p1-N;
35. rewind(fp);
36. fclose(fp);
37.
38. fpn = fopen("MEMO.txt", "wb"); // Vider le fichier initial
39. fclose(fpn);
40.
41. fpn = fopen("MEMO.txt", "w+");
42. for(i=0; i<N; i++,p++){
43.     fwrite(tr, sizeof(struct train), N, fpn);
44. } // Remplir le fichier initial
45. p=p-N;
46. rewind(fpn);
47. fclose(fpn);
48.
49.
50. misajour();

```

Afin de mettre à jour la valeur de EOA en temps réel, nous appellerons la sous-fonction du fichier.

```

1. void misajour( )
2. {
3.     int find = 0;
4.     int END = 0;
5.     char buf[MAXCAR];
6.     printf("Mettre à jour sa EOA d'un trains :\n");
7.     //gets(buf);
8.
9.     //printf("Veuillez saisir la nouvelle EOA:\n");
10.    //scanf("%d", &END);
11.
12.    FILE* fp = fopen("MEMO.txt", "r"); //Fichier initial
13.    FILE* fpn;
14.
15.    p1=tr1;
16.    p=tr;
17.
18.    int m;
19.    int i;
20.    int min = 100;
21.    int max;
22.    int flag;
23.
24.    fread(tr, sizeof(struct train), N, fp);
25.    fread(tr1, sizeof(struct train), N, fp);
26.
27.    for(i=0; i<N; i++,p1++)
28.    {
29.        if(p1->num != 0){
30.            min = 100;
31.            //min = p->position - p1->position;
32.            for(m=0; m<N; m++,p++)
33.            {
34.                if(min > (p->position - p1->position) && (p->position - p1->position) > 0)
35.                {

```

```

36.     min = p->position - p1->position;
37.     p1->End = p->position - 1;
38. }
39. }
40. p = p - N;
41. //p1->End=min - 1;
42. }
43. }
44.
45. p1=p1-N;
46.
47. max = p1->position;
48. p1 = p1 + 1;
49. for(i = 0; i < N; i++,p1++)
50. {
51.     if(p1->position > max)
52.     {
53.         {
54.             max = p1->position;
55.             flag = i;
56.         }
57.     }
58.
59. p1 = p1-N;
60.
61. p1 = p1 + flag;
62. p1->End = 100;
63.
64. p1 = p1 - flag;
65.
66. rewind(fp);
67. fclose(fp);
68.
69. fpn = fopen("MEMO.txt", "wb"); // Vider le fichier initial
70. fclose(fpn);
71.
72. p1--;
73.
74. fpn = fopen("MEMO.txt", "w+");
75. for(i=0; i<N; i++,p1++){
76.     if(p1->num !=0)
77.         printf("%s\t\t%d\t\t%d\n", p1->ID, p1->position, p1->End);
78.     fwrite(tr1, sizeof(struct train), N, fpn);
79. }
80. }
81. p1=p1-N;
82. rewind(fpn);
83. fclose(fpn);
84. }

```

Le programme serveur_train_rev peut faire la comparaison entre la localisation d'un train qui fait l'enquête et les localisations des train qui existent dans la ligne.

```

1. do
2. {
3.     flag = 0; // Initialiser l'autorisation de mouvement
4.
5.     taille=sizeof(adrecliv);
6.
7.     inforecu=recvfrom(sd, bufflect, MAXCAR, 0,(struct sockaddr *) &adrecliv,(socklen_t *)&taille);
8.
9.     printf("\n L'adresse IP du client : %s \n", inet_ntoa(adrecliv.sin_addr));
10.
11. if (inforecu)
12. printf("\n La position recu par le serveur : '%s' \n", bufflect);
13.

```

```

14. else printf("\n Il y a un probleme !!!\n");
15. posi=atoi(bufflect);
16.
17. if(posi == 0)
18. {
19.     printf("AU REVOIR !!\n");
20.     break;
21. }
22.
23. printf("\n La position de ce train est %d: \n",posi);
24.
25. strcpy(buff,"Bien reçu");
26. infoemis=sendto(sd,buff, strlen(buff)+1,0, (const struct sockaddr *) &adrecriv, sizeof(adrecriv));
27.
28. /*
29. taille=sizeof(adrecriv);
30. infoecu=recvfrom(sd, bufflect, MAXCAR, 0,(struct sockaddr *) &adrecriv,(socklen_t *)&taille);
31.
32. printf("\n Adresse IP du client : %s \n", inet_ntoa(adrecriv.sin_addr));
33.
34. if (infoecu)
35. printf("\n La EOA reçu par le serveur : '%s' \n", bufflect);
36.
37. else printf("\n Il y a un probleme !!!\n");
38. eoa=atoi(bufflect);
39. printf("\n La EOA de ce train est %d: \n",eoa);
40.
41. printf("\nMessage emis \n");
42. */
43.
44.
45. if( (fp=fopen("MEMO.txt", "r")) == NULL ){
46.     printf("Erreur sur l'ouverture de fichier, veuillez vous sortir! \n");
47.     getchar();
48.     exit(1);
49. }
50. n=posi;
51.
52. fread(tr1, sizeof(struct train), N, fp);
53. printf("\nN° de train   Localisation   EOA\n") ;
54.
55. for(i=0; i<N; i++,p1++){
56.     if(p1->num == 0) continue;
57.     else
58.     {
59.         printf("%s\t\t%d\t\t%d\n", p1->ID, p1->position, p1->End);
60.         if(0==(p1->position-n))
61.             flag = 2;
62.
63.         else if(0<(p1->position-n)&&(p1->position-n)<2)
64.             flag = 1;
65.         else continue;
66.     }
67.     /* comparaison parmi la fin d'autorisation de mouvement, la localisation d'un train qui fait l'enq
        uête et les localisations des train qui existent dans la ligne */
68. }
69. printf("\n");
70. p1=p1-N;
71. rewind(fp);
72. fclose(fp);
73.
74. if(flag==1)
75.     strcpy(buff,"Passage interdit");
76. else if(flag==2)
77.     strcpy(buff,"Un seul train par canton.");
78. else
79.     strcpy(buff,"Passage permit");
80.
81. infoemis=sendto(sd,buff, strlen(buff)+1,0, (const struct sockaddr *) &adrecriv, sizeof(adrecriv))
;

```

```

82.
83. }
84.
85. while (strcmp(bufflect, "fin") &&strcmp(buff, "fin"));

```

Une application est construite qui teste chacune des fonctions précédentes. On pourra s'appuyer sur un menu afin de permettre à l'utilisateur de tester les fonctions dans l'ordre qu'il souhaite.

Rappel : Faut changer le chemin du fichier selon votre chemin actuelle du fichier. Étant donné que notre environnement d'exécution est macOS, celui qui existe les différences avec Linux, dans ce cas, il faut changer

system("open -a Terminal.app ~/*"); en system("gnome-terminal -e ./test");**

```

1. #include<stdlib.h>
2. #include<stdio.h>
3.
4. int main()
5. {
6.     int choix = 0;
7.     printf("Initialiser l'information de train: \n");
8.     system("open -
a Terminal.app ~/Desktop/STA/gestion_trains_par_RBC_HUO_ZHANG/serveur_train_enreg_fin");
9.     system("open -
a Terminal.app ~/Desktop/STA/gestion_trains_par_RBC_HUO_ZHANG/client_train_enreg_fin");
10.    while(1)
11.    {
12.        printf("1. La suppression d'un train d'un RBC: \n");
13.        printf("2. La mise à jour de la position du train: \n");
14.        printf("3. La lecture de la fin d'autorisation de mouvement d'un train: \n");
15.        printf("4. La mise à jour de la fin d'autorisation de mouvement d'un train: \n");
16.        printf("5. L'enquête d'autorisation d'un nouveau train: \n");
17.
18.        printf("Veuillez vous saisir votre numero du choix: \n");
19.        scanf("%d", &choix);
20.        switch(choix)
21.        {
22.            case 1:
23.                system("open -
a Terminal.app ~/Desktop/STA/gestion_trains_par_RBC_HUO_ZHANG/sup_train_fin");
24.
25.                break;
26.            case 2:
27.                system("open -
a Terminal.app ~/Desktop/STA/gestion_trains_par_RBC_HUO_ZHANG/mis_a_jour_local_fin");
28.                break;
29.            case 3:
30.                system("open -
a Terminal.app ~/Desktop/STA/gestion_trains_par_RBC_HUO_ZHANG/Lecture_train_EOA");
31.                break;
32.            case 4:
33.                system("open -
a Terminal.app ~/Desktop/STA/gestion_trains_par_RBC_HUO_ZHANG/mis_a_jour_eoa_fin");
34.                break;
35.            case 5:
36.                system("open -
a Terminal.app ~/Desktop/STA/gestion_trains_par_RBC_HUO_ZHANG/serveur_train_rev");
37.                system("open -
a Terminal.app ~/Desktop/STA/gestion_trains_par_RBC_HUO_ZHANG/client_train_rev");
38.                break;
39.        }
40.
41.        printf("Veuillez vous sortir? 0: Sortez, 1: Restez\n");

```

```

42.     int fin;
43.     scanf("%d", &fin);
44.     if(fin == 0) break;
45. }
46.
47. }

```

4. Exécution des programmes.

D'abord, le programme Test_general devrait être exécuté.

```

/Users/huojiaxi/Desktop/gestion_trains_par_RBC_HUO_ZHANG/Test_general ; exit;
(base) MacBook-Pro:~ huojiaxi$ /Users/huojiaxi/Desktop/gestion_trains_par_RBC_HUO_ZHANG/Test_general ; exit;
Initialiser l'information de train:
1. La suppression d'un train d'un RBC:
2. La mise à jour de la position du train:
3. La lecture de la fin d'autorisation de mouvement d'un train:
4. La mise à jour de la fin d'autorisation de mouvement d'un train:
5. L'enquête d'autorisation d'un nouveau train:
Veuillez vous saisir votre numero du choix:

```

En même temps, les programmes serveur_train_enreg et client_train_enreg seront exécutés simultanément afin d'initialiser la ligne et sauvegarder les informations initiales :

```

(base) MacBook-Pro:~ huojiaxi$ /Users/huojiaxi/Desktop/gestion_trains_par_RBC_HUO_ZHANG/serveur_train_enreg ; exit;
Appuyez sur la touche Entrée pour commencer

Attente la reponse du serveur

/Users/huojiaxi/Desktop/gestion_trains_par_RBC_HUO_ZHANG/client_train_enreg ; exit;
(base) MacBook-Pro:~ huojiaxi$ /Users/huojiaxi/Desktop/gestion_trains_par_RBC_HUO_ZHANG/client_train_enreg ; exit;
Appuyez sur la touche Entrée pour commencer

warning: this program uses gets(), which is unsafe.
Envoyez votre identifiant :|

```

```

TGV809      15

Adresse IP du client : 127.0.0.1

L'identifiant reçu par le serveur : 'fin'
AU REVOIR !!

N° de train  Localisation  EOA
TGV134      67            69
TGV083      43            66
TER203      21            42
TER504      70            100
TGV809      15            20

Terminer ce programme qui enregistre l'information de train!!!
logout
Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Opération terminée]

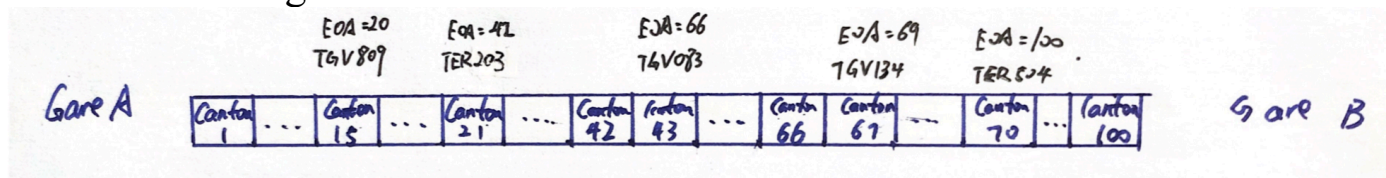
```

Comme indiqué dans la figure ci-dessus, vous pourriez trouver que les informations sont bien enregistrées des 3 trains sur la ligne

	N° de train	Localisation	EOA
1	TGV134	67	100
2	TGV083	43	66
3	TER203	21	42

Et 2 plus trains sont enregistrées afin de tester les fonctions au futur. Dans cet exemple, nous mettons 5 trains TGV134, TGV083, TER203, TER504 et TGV809 sur la ligne. En réalité, notre système permet mettre 100 trains sur la ligne, de plus, notre système permet mettre plus de trains en modifiant le paramètre N dans les scripts. Ses EOAs sont gérés automatiquement.

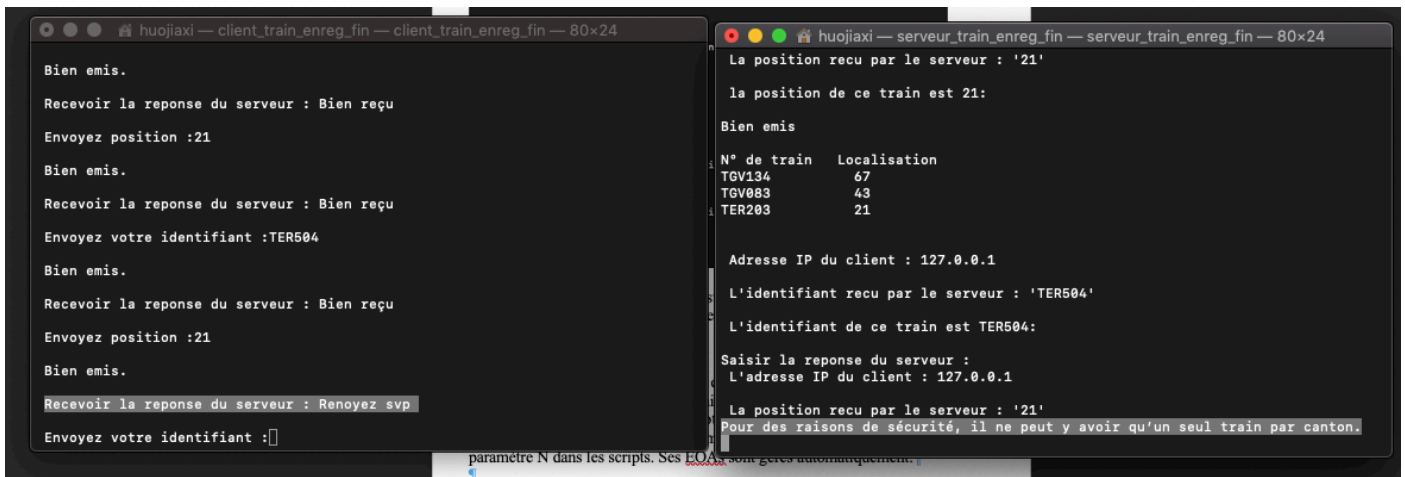
L'état de cette ligne :



Pour choisir la fonction à tester, justement saisissez le numéro de fonction, la fonction choisie va être exécuté automatiquement :

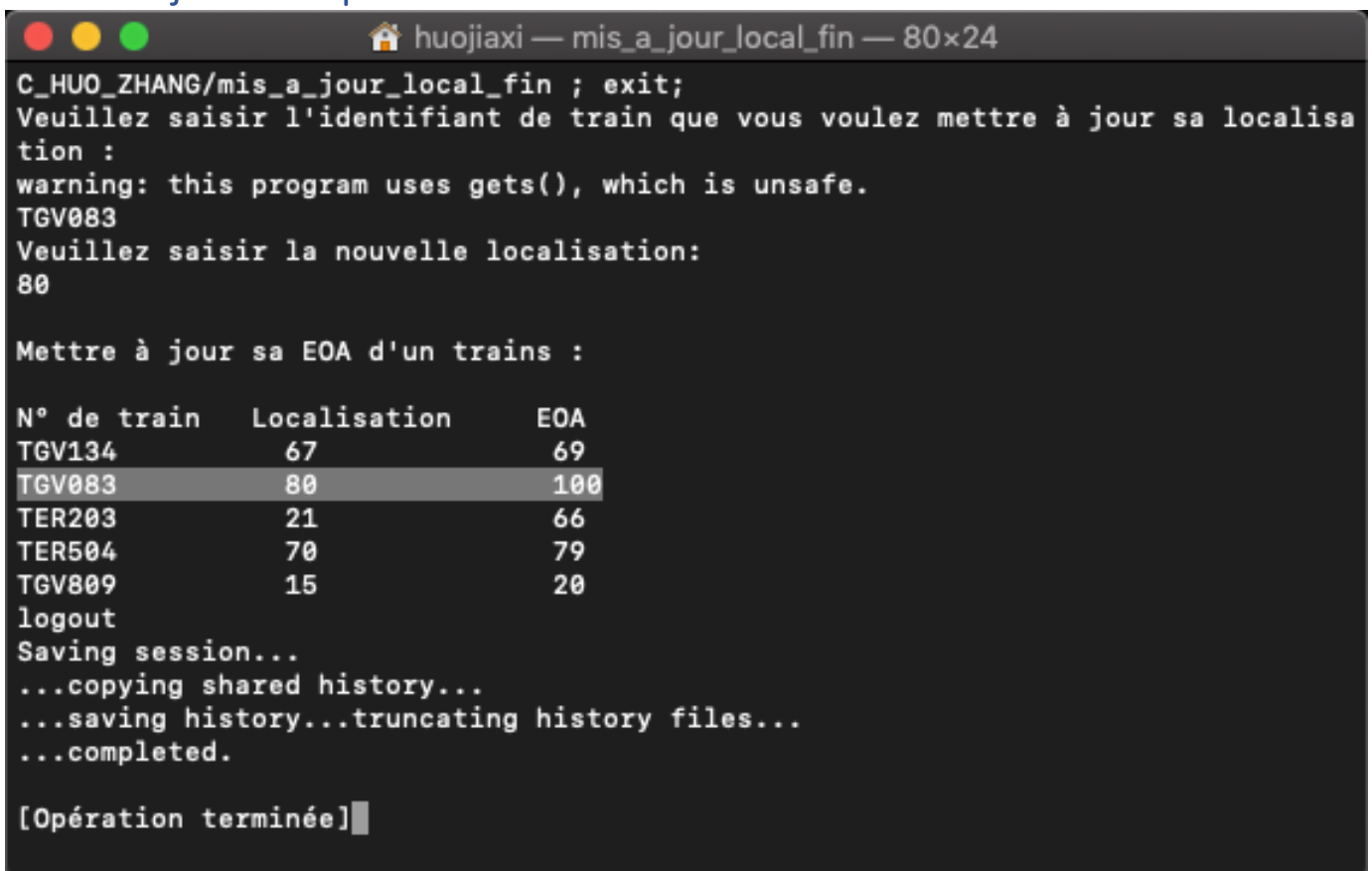
Rappel :

Une ligne ferroviaire est divisée en cantons de 1,5 km. Pour des raisons de sécurité, il ne peut y avoir qu'un seul train par canton. Donc notre système ne permet pas mettre 2 trains dans le même canton :



Cette opération est bien interdite.

La mise à jour de la position du train



Comme indiqué dans la figure ci-dessus, vous pourriez trouver que la localisation de train TGV134 est bien mise à jour. De 43 à 80. Les EOA des trains sur la ligne sont mises à jour en même temps.


```

/Users/huojiaxi/Desktop/STA/gestion_trains_par_RBC_HUO_ZHANG/mis_a_jour_local_fin ; exit;
(base) MacBook-Pro:~ huojiaxi$ /Users/huojiaxi/Desktop/STA/gestion_trains_par_RBC_HUO_ZHANG/mis_a_jour_local_fin ; exit;
Veuillez saisir l'identifiant de train que vous voulez mettre à jour sa localisation :
warning: this program uses gets(), which is unsafe.
TGV809
Veuillez saisir la nouvelle localisation:
67

Pour des raisons de sécurité, il ne peut y avoir qu'un seul train par canton.
Collision avec le train : TGV134

logout
Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Opération terminée]

```

Pour des raisons de sécurité, il ne peut y avoir qu'un seul train par canton. Si nous actualisons la position du TGV809 sur 67, le système émettra un avertissement, car 67 correspondra à l'emplacement du TGV134, cette opération sera désactivée et le système indiquera le train qui va entrer en collision.

La mise à jour de la fin d'autorisation de mouvement d'un train.

```

| | | | | ( |      | _ _ _ | ! |
| _ _ _ _ _       | _ _ _ _ _   ! |
| | | | |         | _ | _ |     |
.' \ \ , \ \ _    / _ / _ _ |
/Users/huojiaxi/Desktop/STA/gestion_trains_par_RBC_HUO_ZHANG/mis_a_jour_eoa_fin
; exit;
(base) MacBook-Pro:~ huojiaxi$ /Users/huojiaxi/Desktop/STA/gestion_trains_par_RB
C_HUO_ZHANG/mis_a_jour_eoa_fin ; exit;
Mettre à jour sa EOA d'un trains :

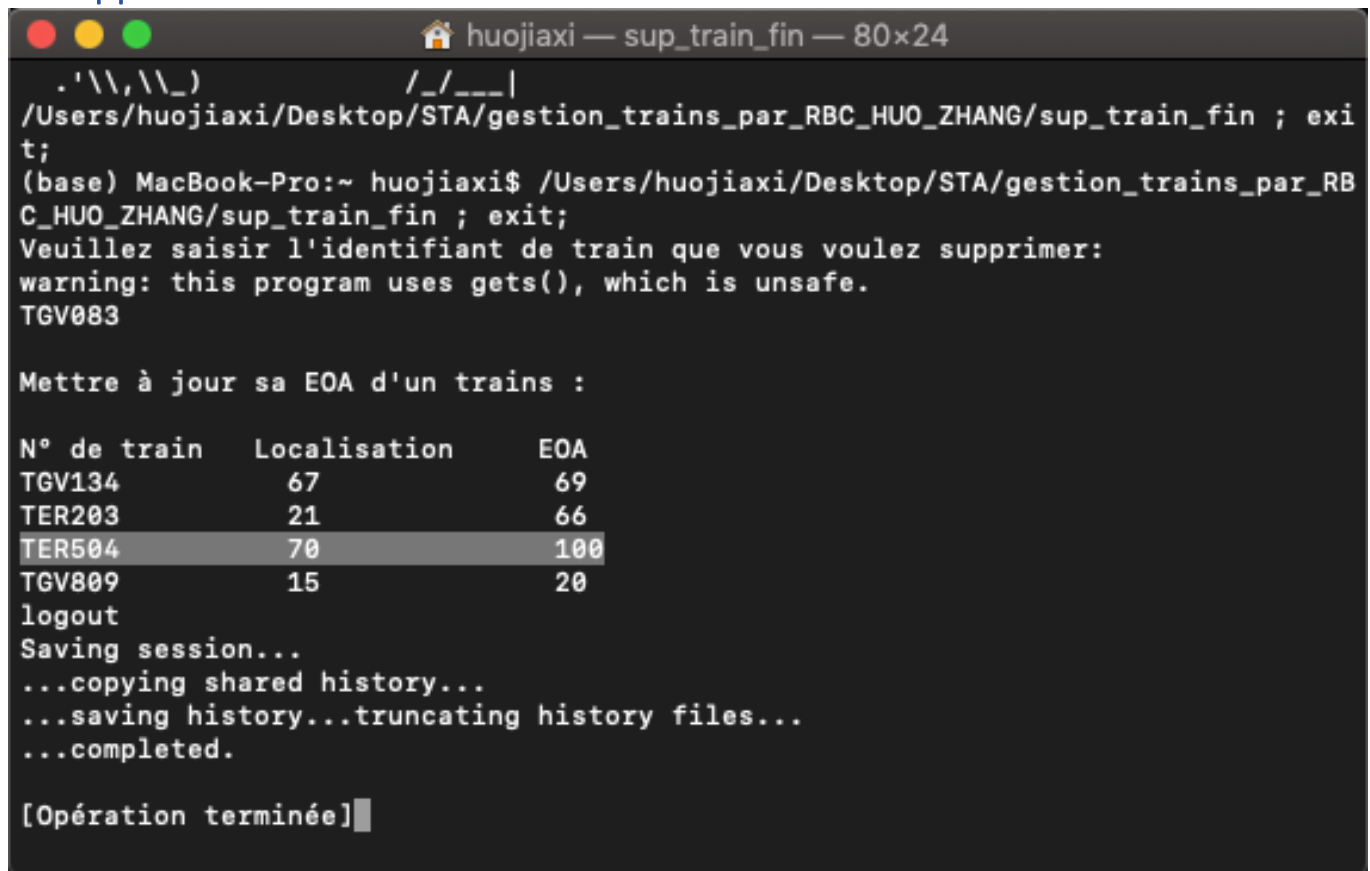
N° de train   Localisation   EOA
TGV134        67             69
TGV083        80             100
TER203        21             66
TER504        70             79
TGV809        15             20
logout
Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Opération terminée]

```

Comme indiqué dans la figure ci-dessus, vous pourriez trouver que la fin d'autorisation de mouvement de train est bien mise à jour.

La suppression d'un train d'un RBC



```
. '\, \_) /_/__|
/Users/huojiaxi/Desktop/STA/gestion_trains_par_RBC_HUO_ZHANG/sup_train_fin ; exit;
(base) MacBook-Pro:~ huojiaxi$ /Users/huojiaxi/Desktop/STA/gestion_trains_par_RBC_HUO_ZHANG/sup_train_fin ; exit;
Veuillez saisir l'identifiant de train que vous voulez supprimer:
warning: this program uses gets(), which is unsafe.
TGV083

Mettre à jour sa EOA d'un trains :

N° de train   Localisation   EOA
TGV134        67             69
TER203        21             66
TER504        70             100
TGV809        15             20
logout
Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Opération terminée]
```

Nous supposons que le train TGV083 arrivait à la gare B, donc il devrait être supprimé de cette ligne.

Comme indiqué dans la figure ci-dessus, vous pourriez trouver que le train TGV083 est bien supprimé, il reste 4 trains sur la ligne maintenant : TGV134, TER203, TER504 et TGV809. Vous pourriez trouver que les EOAs sont bien mises à jour. Le train TER504 devient le train qui est le plus proche de la gare B.

La lecture de la fin d'autorisation de mouvement d'un train

```

\ _ ~ \ _ | ||      \ _   |___|
 /~\  \_    ___/     \_   /___|
|||/||~~|           .'. '~\
|||||---|          |__|_|_|
|||||(|         |___|_|_|
|-----'\       |___|\_/_
|_|_|_|         |_|_|_|
.'\\,\,\_)      /_/___|

/Users/huojiaxi/Desktop/STA/gestion_trains_par_RBC_HUO_ZHANG/Lecture_train_EOA ;
exit;
(base) MacBook-Pro:~ huojiaxi$ /Users/huojiaxi/Desktop/STA/gestion_trains_par_RB
C_HUO_ZHANG/Lecture_train_EOA ; exit;
La lecture de EOA pour le train:
warning: this program uses gets(), which is unsafe.
TGV134
La fin d'autorisation de mouvement de ce train: 69
logout
Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Opération terminée]

```

Comme indiqué dans la figure ci-dessus, vous pourriez trouver que la fin d'autorisation de mouvement de train TGV134 est bien indiquée : 69.

L'enquête d'autorisation d'un nouveau train

[illegible]

Comme indiqué dans la figure ci-dessus, pour des raisons de sécurité, il ne peut y avoir qu'un seul train par canton. Si le système RBC reçoit la même localisation que le train existant dans la ligne, le système enverra une réponse au train pour indiquer le chauffeur qu'il ne peut y avoir qu'un seul train par canton.

```

serveur_train_rev:
(base) MacBook-Pro:~ huojiaxi$ /Users/huojiaxi/Desktop/STA/gestion_trains_par_RB
C_HUO_ZHANG/serveur_train_rev ; exit;
Appuyez sur la touche Entrée pour commencer

Attente la reponse du serveur !!!

L'adresse IP du client : 127.0.0.1

La position reçu par le serveur : '14'

La position de ce train est 14:

N° de train  Localisation  EOA
TGV134      67           69
TER203      21           66
TER504      70          100
TGV809      15           20

L'adresse IP du client : 127.0.0.1

client_train_rev:
(base) MacBook-Pro:~ huojiaxi$ /Users/huojiaxi/Desktop/STA/gestion_trains_par_RB
C_HUO_ZHANG/client_train_rev ; exit;
Appuyez sur la touche Entrée pour commencer

warning: this program uses gets(), which is unsafe.
Envoyez votre position :14

Bien emis!

Recevoir la reponse du serveur : Bien reçu
Recevoir la reponse du serveur : Passage interdit

```

Comme indiqué dans la figure ci-dessus, si un train est au canton 14, il peut recevoir la réponse de RBC : **Passage interdit**, c'est-à-dire il n'a pas d'autorisation de mouvement, il faut le ralentir immédiatement pour éviter une collision avec le TGV809 en face.

```

serveur_train_rev:
La position reçu par le serveur : '14'

La position de ce train est 14:

N° de train  Localisation  EOA
TGV134      67           69
TER203      21           66
TER504      70          100
TGV809      15           20

L'adresse IP du client : 127.0.0.1

La position reçu par le serveur : '16'

La position de ce train est 16:

N° de train  Localisation  EOA
TGV134      67           69
TER203      21           66
TER504      70          100
TGV809      15           20

client_train_rev:
warning: this program uses gets(), which is unsafe.
Envoyez votre position :14

Bien emis!

Recevoir la reponse du serveur : Bien reçu
Recevoir la reponse du serveur : Passage interdit

Envoyez votre position :16

Bien emis!

Recevoir la reponse du serveur : Bien reçu
Recevoir la reponse du serveur : Passage permit

```

Comme indiqué dans la figure ci-dessus, si un train est au canton 16, il peut recevoir la réponse de RBC : **Passage permit**, c'est-à-dire il a l'autorisation de mouvement. La distance entre lui et le train devant lui est en sécurité.

Quitter l'application

```

Test_general:
Veuillez vous saisir votre numero du choix:
1
Veuillez vous sortir? 0: Sortez, 1: Restez
1
1. La suppression d'un train d'un RBC:
2. La mise à jour de la position du train:
3. La lecture de la fin d'autorisation de mouvement d'un train:
4. La mise à jour de la fin d'autorisation de mouvement d'un train:
5. L'enquête d'autorisation d'un nouveau train:
Veuillez vous saisir votre numero du choix:
5
Veuillez vous sortir? 0: Sortez, 1: Restez
0
logout
Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

```

Comme indiqué ci-dessus, notre application fonctionne bien et permet l'utilisateur tester toutes les fonctions dans l'ordre qu'il souhaite.

5. Bilan :

Au cours de ce travail, nous avons profondément compris le principe de contrôle du signal des trains et conçu le système de contrôle des trains à la main, ce qui nous a permis d'approfondir la compréhension du contrôle des trains et d'améliorer les capacités de programmation du langage C.