```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <signal.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <string.h>
#include <semaphore.h>
#include <errno.h>
#include"header.h"

#define PORT1 "55558"


int main(int argc, char **argv){

//******variables*****
struct requete structure;

//creation de la memoire partagée
int* shmem = shm_creation(128);
memcpy(shmem, spect, sizeof(spect));

//initialisation structure
init_struct(&structure);
//initialisation du sémaphore
sem_init(&semaphore, 1, 1);


//******config socket*****************
int sock_fd, srvc_fd;
struct addrinfo s_init, *servinfo, *p;
struct sockaddr_storage client_addr;
socklen_t s_taille;

memset(&s_init, 0, sizeof(s_init));
s_init.ai_family = AF_UNSPEC;
s_init.ai_socktype = SOCK_STREAM;
s_init.ai_flags = AI_PASSIVE;
if (getaddrinfo(NULL, PORT1, &s_init, &servinfo) != 0) {
fprintf(stderr, "Erreur getaddrinfo\n");
exit(1);
}
for(p = servinfo; p != NULL; p = p->ai_next) {
if ((sock_fd = socket(p->ai_family, p->ai_socktype, p->ai_protocol)) == -1) {
perror("Serveur: socket");
continue;
}
if (bind(sock_fd, p->ai_addr, p->ai_addrlen) == -1) {
close(sock_fd);
perror("Serveur: erreur bind");
continue;
}
break;
}
if (p == NULL) {
fprintf(stderr, "Serveur: echec bind\n");
exit(2);
}
freeaddrinfo(servinfo);
if (listen(sock_fd, 5) == -1) {
        perror("listen");
```

```
exit(1);
}


// signal pour ignorer les fils
signal(SIGCHLD, SIG_IGN);

//******************************************

while(1){

s_taille = sizeof(client_addr);
srvc_fd = accept(sock_fd, (struct sockaddr *) &client_addr, &s_taille);

if (srvc_fd == -1) {
perror("accept");
continue;
}


printf("Nouvelle requete recue.\n");

        if(fork() == 0){

        close(sock_fd);
        //reception de la requete
        recv(srvc_fd, &structure, sizeof(struct requete), 0);

        //si reservation
        if(structure.type == 1){

                if(fork()==0){
                        // On attend la disponibilité du sémaphore
                        sem_wait(&semaphore);
                        if((shmem[structure.spectacles]-structure.nb_place_resa)>=0){
                                shmem[structure.spectacles]=shmem[structure.spectacles]-
structure.nb_place_resa;
                                structure.nb_place_dispo=shmem[structure.spectacles];
                                structure.confirm_resa=1;
                        }else{
                                structure.nb_place_dispo=shmem[structure.spectacles];
                                structure.confirm_resa=0;
                        }
                        // On relache le sémaphore
                        sem_post(&semaphore);

                        send(srvc_fd, &structure, sizeof(struct requete), 0);

                        close(srvc_fd);
                        exit(0);
                }
        }
        //si consultation
        if(structure.type == 0){
                if(fork()==0){
                        //renseignement du nombre de places
                        structure.nb_place_dispo=shmem[structure.spectacles];
                        send(srvc_fd, &structure, sizeof(struct requete), 0);

                        close(srvc_fd);
                        exit(0);
                }

        }

        close(srvc_fd);
```

```
        exit(0);
        }


}

//sem_destroy(&semaphore);
close(sock_fd);
exit(0);
}
```