

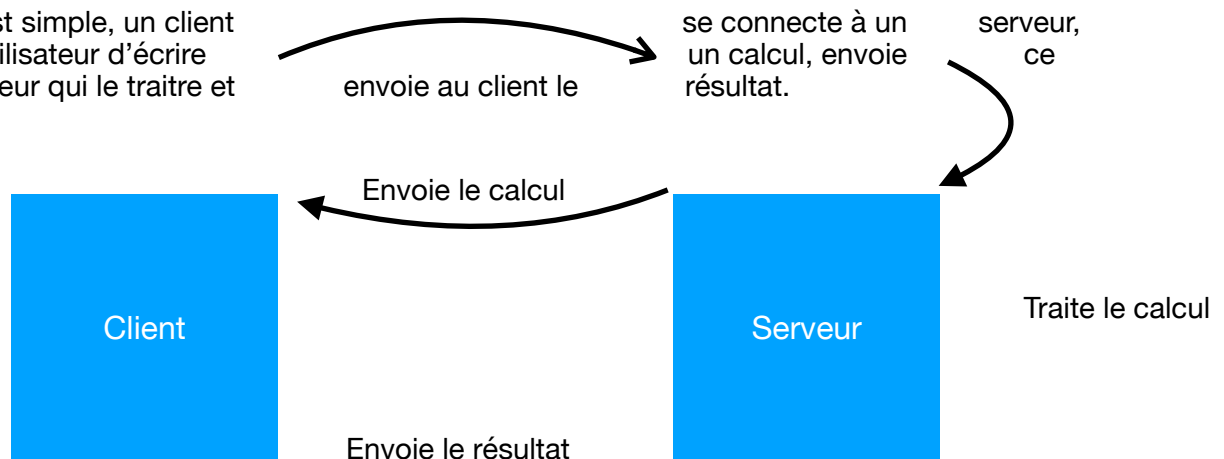
DEVELOPPEMENT APPLICATION CALCULATRICE

DEVELOPPEMENT APPLICATION CALCULATRICE	1
I. INTRODUCTION	3
II. CLIENT C	3
III. SERVEUR C	5
A. Serveur C de Lecture (V0)	5
B. Serveur C de calcul (V1)	6
C. Serveur C pour plusieurs clients (V2)	8

I. INTRODUCTION

Ayant fait la calculatrice simple sur mobile, nous allons maintenant tester le développement d'un calculatrice par serveur.

Le concept est simple, un client propose à l'utilisateur d'écrire calcul au serveur qui le traite et



Pour commencer nous allons programmer en C un serveur et un client et ensuite programmer un client en Java.

II. CLIENT C

Nous allons ici développer un simple client en langage C qui ne fera qu'envoyer du texte au serveur.

Celui-ci devra être lancé avec deux arguments, un pour l'adresse IP (127.0.0.1 l'adresse de loopback ici) et un pour le port que l'on choisira plus tard lors du développement du serveur.

Le Client se lancera en tapant par exemple : `./Client 127.0.0.1 5001`
Puis il suffira d'entrer un message pour que celui-ci soit envoyé au serveur.

```
portno = atoi(argv[2]);

/* Create a socket point */
sockfd = socket(PF_INET, SOCK_STREAM, 0);

server = gethostbyname(argv[1]);

//bzero((char *) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
bcopy((char *)server->h_addr, // identique à memcpy
      (char *)&serv_addr.sin_addr.s_addr, server->h_length);

serv_addr.sin_port = htons(portno);
```

```
/* Now connect to the server */
connect(sockfd,(struct sockaddr *)&serv_addr,sizeof(serv_addr));
```

On va donc créer une socket avec les deux arguments entrés (IP + port).

La socket créée, il faut désormais se connecter au serveur.

```
/* Now ask for a message from the user, this message
 * will be read by server
 */
printf("Please enter the message: ");
bzero(buffer,256);
fgets(buffer,255,stdin);
```

Nous allons donc nous servir du « buffer » pour écrire un message destiné au serveur.

```
/* Send message to the server */
n = write(sockfd,buffer,(ssize_t)strlen(buffer));
```

Et donc l'envoyer au serveur.

```
/* Now read server response */
bzero(buffer,256);
n = read(sockfd,buffer,(ssize_t)255);

printf("%s\n",buffer);
```

Normalement, le serveur sera configuré pour envoyer une réponse, nous allons donc ajouter ces lignes permettant de lire cette réponse.

Ne pouvant pas tester un client seul nous allons donc d'abords procéder à l'écriture d'un simple serveur ne procédant qu'à une lecture en envoyant une réponse, puis tester les deux programmes ensemble.

III. SERVEUR C

A. Serveur C de Lecture (V0)

Nous commençons tout d'abord avec un serveur qui reçoit un message puis affiche celui-ci et réponds au client (donc pas de calcul ici).

```
int sockfd, newsockfd, portno;
unsigned int clilen;
char buffer[256];
struct sockaddr_in serv_addr, cli_addr;
int n;

/* First call to socket() function */
sockfd = socket(PF_INET, SOCK_STREAM, 0);

/* Initialize socket structure */
bzero(&serv_addr, sizeof(serv_addr));
portno = 5001;

serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = INADDR_ANY;
serv_addr.sin_port = htons(portno);

/* Now bind the host address using bind() call.*/
bind(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
```

Tout d'abords nous créons une socket (adresse ip associée a un port) pour que le serveur écoute sur le port 5001, le client devra donc utiliser le port 5001 pour communiquer avec le serveur.

```
/* Now start listening for the clients, here process will
 * go in sleep mode and will wait for the incoming connection */
listen(sockfd,5);4
```

Le serveur commence donc a écouter pour les clients.

```
clilen = sizeof(cli_addr);
while (1)
{
    /* Accept actual connection from the client */
    newsockfd = accept(sockfd, (struct sockaddr *)&cli_addr, &clilen);
    /* If connection is established then start communicating */
    bzero(buffer,256); // met le buffer à zéro (identique à memset)
    n = read( newsockfd,buffer,(ssize_t)255 );

    printf("Here is the message: %s\n",buffer);
    /* Write a response to the client */
    n = write(newsockfd,"I got your message",(ssize_t)18);
}
```

Le serveur va donc attendre que un client rentre en contact avec lui puis, quand un client rentrera en contact avec lui, le serveur acceptera la tentative de communication du client, lira son message en l'affichant puis répondra au client par « I got your message ».

À noter que cette version du serveur ne peut gérer qu'un client à la fois.

Nous allons donc tester le serveur et le client :

```
Here is the message: Calc! eat up message text
```

```
lMacclient@Calc:~$ ./clientDeCalculProg 127.0.0.1 5001
Please enter the message: Calc! eat up message text
I got your message
lMacclient@Calc:~$
```

On peut donc voir que le serveur a bien reçu le message du client et a envoyé un message que le client a reçu et affiché.

On passe donc à la suite.

B. Serveur C de calcul (V1)

Nous avons donc développé une nouvelle version du serveur de calcul, la précédente ne faisant que lire le message reçu. Celle-ci sera donc capable de traiter un calcul puis envoyer le résultat.

Ce serveur sera donc pas si différent, nous pouvons garder le même code mais simplement changer la partie avant l'envoi d'une réponse.

```
int calcul(char ent[]) {
    int res = 0;
    if(strstr(ent, "+") != NULL) {
        printf("Il s'agit d'une addition\n");
        char delim[] = "+";
        char *ptr = strtok(ent, delim);
        int num1 = atoi(ptr);
        printf("le premier numéro est : %d\n", num1);
        ptr = strtok(NULL, delim);
        int num2 = atoi(ptr);
        printf("le deuxième numéro est : %d\n", num2);
        res = num1 + num2;
    }
    if(strstr(ent, "-") != NULL) {
        printf("Il s'agit d'une soustraction\n");
        char delim[] = "-";
        char *ptr = strtok(ent, delim);
        int num1 = atoi(ptr);
        printf("le premier numéro est : %d\n", num1);
        ptr = strtok(NULL, delim);
        int num2 = atoi(ptr);
        printf("le deuxième numéro est : %d\n", num2);
        res = num1 - num2;
    }
    if(strstr(ent, "*") != NULL) {
        printf("Il s'agit d'une multiplication\n");
        char delim[] = "*";
        char *ptr = strtok(ent, delim);
        int num1 = atoi(ptr);
        printf("le premier numéro est : %d\n", num1);
        ptr = strtok(NULL, delim);
        int num2 = atoi(ptr);
        printf("le deuxième numéro est : %d\n", num2);
        res = num1 * num2;
    }
    if(strstr(ent, "/") != NULL) {
        printf("Il s'agit d'une division\n");
        char delim[] = "/";
        char *ptr = strtok(ent, delim);
        int num1 = atoi(ptr);
        printf("le premier numéro est : %d\n", num1);
        ptr = strtok(NULL, delim);
        int num2 = atoi(ptr);
        printf("le deuxième numéro est : %d\n", num2);
        res = num1 / num2;
    }
}
```

Nous avons donc écrit une fonction qui permettra de traiter un calcul (nous avons repris l'algorithme de la calculatrice simple).

```
printf("Here is the message: %s", buffer);
sprintf(buffer, "%d", calcul(buffer));
printf("Le resultat est : %s\n\n", buffer);
```

Lorsque l'on reçoit le message on va donc le faire passer dans la fonction pour recevoir le résultat.

```

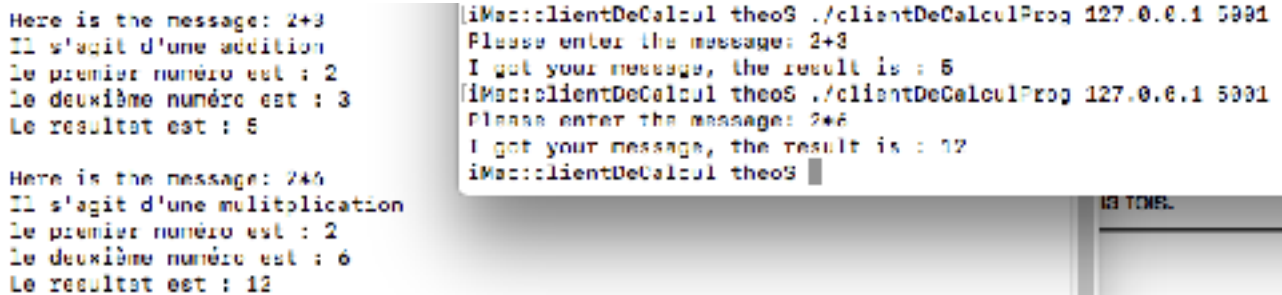
/* Write a response to the client */
char msg[256] = "I got your message, the result is : ";
strcat(msg, buffer);
n = write(newsockfd, msg, strlen(msg));

```

On va donc insérer le résultat dans un message et l'envoyer au client.

Ce serveur est donc capable de donner des résultat a partir de calculs, mais n'est toujours pas capable de traiter plus d'un client à la fois.

On va donc tester le nouveau serveur :



```

Here is the message: 2+3
Il s'agit d'une addition
le premier numéro est : 2
le deuxième numéro est : 3
Le resultat est : 5

Here is the message: 2*6
Il s'agit d'une multiplication
le premier numéro est : 2
le deuxième numéro est : 6
Le resultat est : 12

[iMac:clientDeCalcul theoS ./clientDeCalculProg 127.0.0.1 5991
Please enter the message: 2+3
I got your message, the result is : 5
[iMac:clientDeCalcul theoS ./clientDeCalculProg 127.0.0.1 5991
Please enter the message: 2*6
I got your message, the result is : 12
iMac:clientDeCalcul theoS

```

Le client envoie un calcul au serveur, le serveur effectue ce calcul puis envoie le résultat au client.

Nous allons donc procéder a l'écriture d'un serveur pour traiter plusieurs clients.

C. Serveur C pour plusieurs clients (V2)

Nous allons donc désormais développer un serveur capable de gérer plusieurs clients.

```
void doprocessing (int sock){
    int n;
    char buffer[256];
    bzero(buffer,256);

    n = read(sock,buffer,255);
    printf("Here is the message: %s\n",buffer);

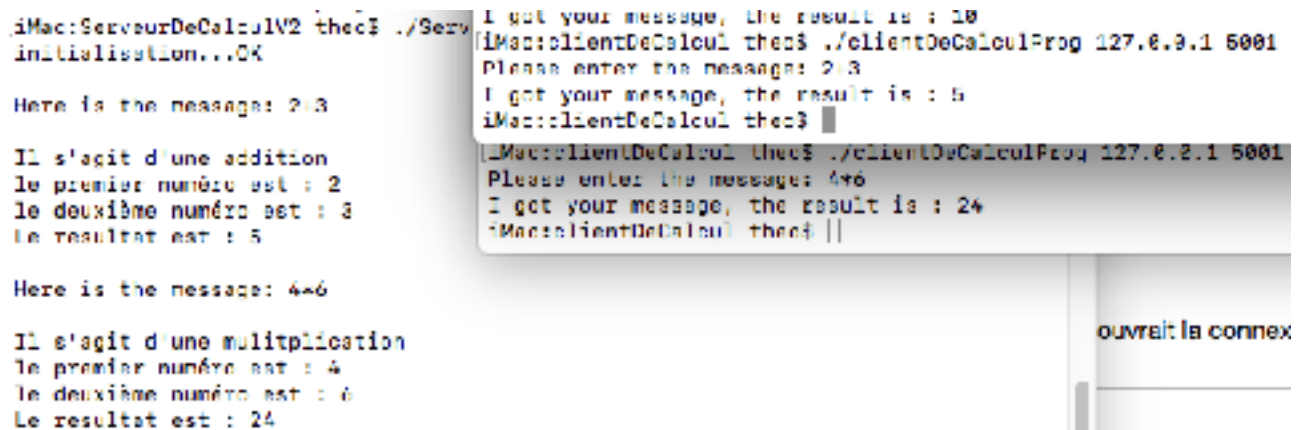
    sprintf(buffer, "%d", calcul(buffer));
    printf("Le resultat est : %s\n\n",buffer);
    char msg[256] = "I got your message, the result is : ";
    strcat(msg, buffer);
    n = write(sock, msg, strlen(msg));
    //n = write(sock,"I got your message",18);
}
```

Tout d'abords on crée une fonction qui permet de lire, traiter et répondre.

```
pid = fork();
if (pid == 0){
    /* This is the client process */
    close(sockfd);
    doprocessing(newsockfd);
    exit(0);
}else{
    close(newsockfd);
}
```

Puis dans la précédente boucle qui ouvrait la connexion, on va mettre une condition qui traite le client et puis ferme la connexion.

Nous allons donc tester le serveur V2 :



The screenshot shows two terminal windows. The left window is the server's terminal, and the right window is the client's terminal.

Server Terminal (iMac:ServeurDeCalculV2):

```
iMac:ServeurDeCalculV2 thec$ ./ServeurDeCalculProg
initialisation...OK

Here is the message: 2+3

Il s'agit d'une addition
le premier numéro est : 2
le deuxième numéro est : 3
le resultat est : 5

Here is the message: 4*6

Il s'agit d'une multiplication
le premier numéro est : 4
le deuxième numéro est : 6
Le resultat est : 24
```

Client Terminal (iMac:clientDeCalcul):

```
iMac:clientDeCalcul thec$ ./clientDeCalculProg 127.0.0.1 5001
Please enter the message: 2+3
I got your message, the result is : 5
iMac:clientDeCalcul thec$

iMac:clientDeCalcul thec$ ./clientDeCalculProg 127.0.0.1 5001
Please enter the message: 4*6
I got your message, the result is : 24
iMac:clientDeCalcul thec$
```

On the right side of the client terminal, there is a button labeled "ouvrir la connexion".

Le serveur reçoit donc bien les deux calcul et les renvoie aux bons clients.
