Note technique

Binômes : WONG Hoe Ziet, BINTI MOHAMAD Nurdini

Les choix techniques

- ❖ Langage de programmation pour les scripts : langage C

  - ➢ Le langage de programmation le plus rapide et performant.
  - ➢ Offre des fonctions qui permettent un accès simple et direct aux fonctions internes de l'ordinateur (ex : la gestion de la mémoire).
  - ➢ Pas besoin d'interpréter ou d'un IDE comme Python et Java
  - ➢ Compiler GCC natif à Linux, pas besoin de l'installer
  - ➢ Beaucoup de ressources disponibles en ligne

- ❖ Langage utilisé pour l'interface web : HTML

  - ➢ Le langage de programmation web le plus simple à manipuler
  - ➢ Simple à effectuer des traitements de données (affichage des données) à l'aide de langage C

- ❖ Choix de stream de données en socket : TCP

  - ➢ Nous avons besoins d'assurer la fiabilité des données et les pertes de données ne sont pas tolérées, afin que l'administrateur puisse analyser les données en détail.

- ❖ Mode de fonctionnement permettant connexion de plusieurs clients au serveur : fork ()

  - ➢ Choix entre fork() et thread()
  - ➢ Méthode fork() est plus simple à mettre en place, peu de lignes de code à ajouter
  - ➢ Plus adapté pour notre architecture, car chaque connexion est lancée sur un différent processus et possède sa propre espace mémoire dédiée
  - ➢ En revanche, au niveau de performance, fork risque d'être plus lente que thread dans un système de multiprocesseurs

❖ Architecture pour monitorer parc machines

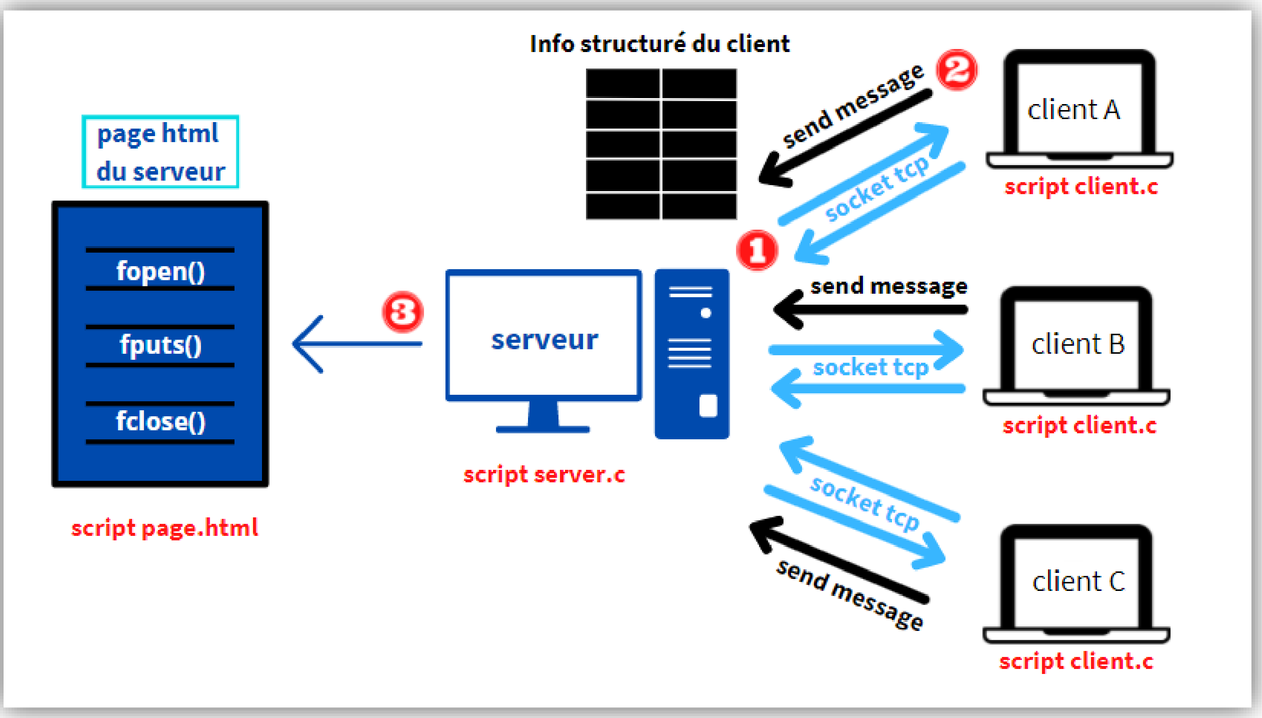
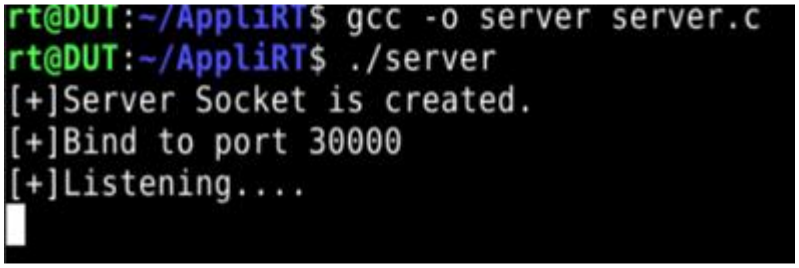


Figure 1 : Le schéma d'architecture réseau

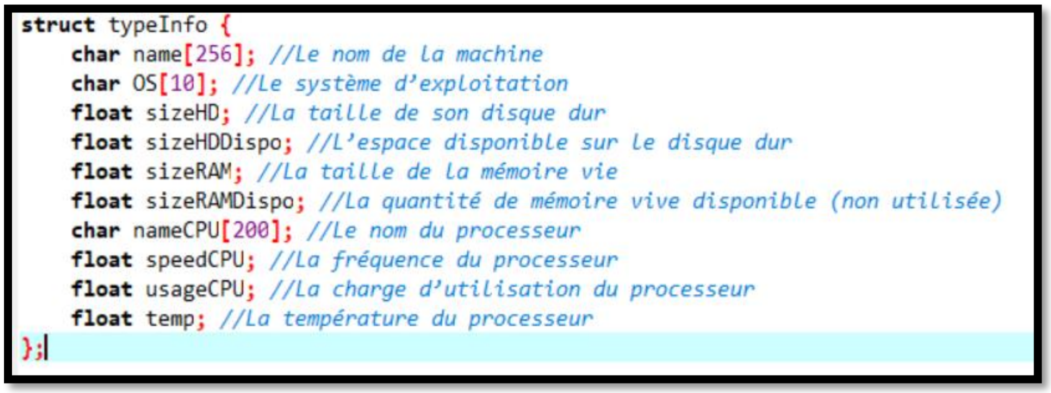
1. Établissement de **connexion tcp entre client et serveur** en utilisant le **socket**.



```
rt@DUT:~/AppliRT$ gcc -o server server.c
rt@DUT:~/AppliRT$ ./server
[+]Server Socket is created.
[+]Bind to port 30000
[+]Listening....
```

Figure 2 : Le serveur ouvre la connexion

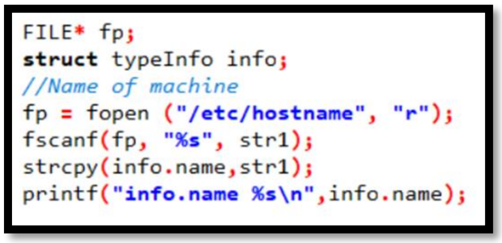
2. Client **envoie les informations** dans le format d'une structure **chaque 10s**.
   a. Défini la structure des informations sur le client :



```
struct typeInfo {
    char name[256]; //Le nom de la machine
    char OS[10]; //Le système d'exploitation
    float sizeHD; //La taille de son disque dur
    float sizeHDDispo; //L'espace disponible sur le disque dur
    float sizeRAM; //La taille de la mémoire vie
    float sizeRAMDispo; //La quantité de mémoire vive disponible (non utilisée)
    char nameCPU[200]; //Le nom du processeur
    float speedCPU; //La fréquence du processeur
    float usageCPU; //La charge d'utilisation du processeur
    float temp; //La température du processeur
};
```

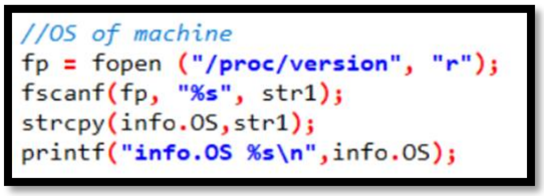
Figure 3 : Le structure d'information de client

   b. Le client cherche et affect la valeur pour chaque information :



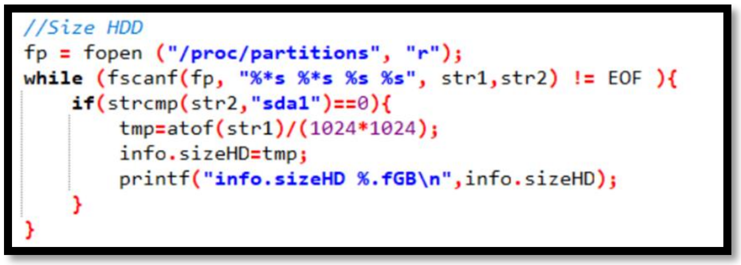
```
FILE* fp;
struct typeInfo info;
//Name of machine
fp = fopen ("/etc/hostname", "r");
fscanf(fp, "%s", str1);
strcpy(info.name,str1);
printf("info.name %s\n",info.name);
```

Figure 4 : Récupération et d'effectuation du nom du client sur la structure



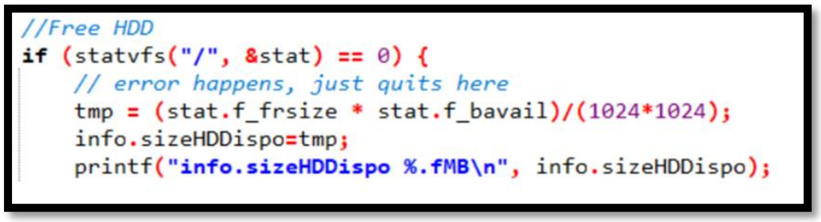
```
//OS of machine
fp = fopen ("/proc/version", "r");
fscanf(fp, "%s", str1);
strcpy(info.OS,str1);
printf("info.OS %s\n",info.OS);
```

Figure 5 : Récupération et d'effectuation d'OS du client sur la structure



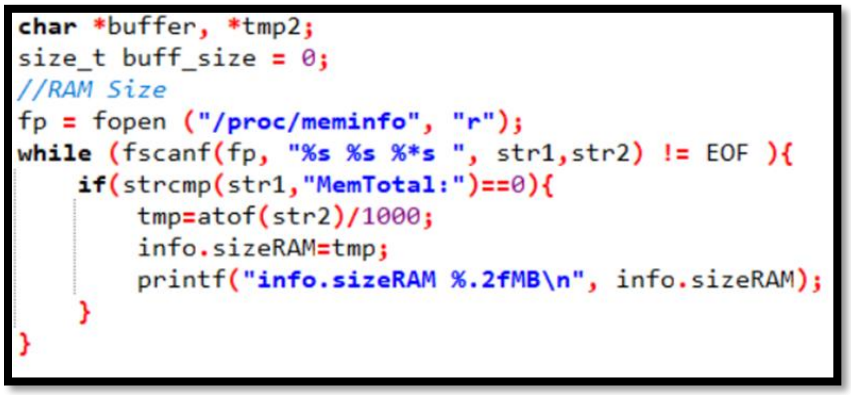
```
//Size HDD
fp = fopen ("/proc/partitions", "r");
while (fscanf(fp, "%*s %*s %s %s", str1,str2) != EOF ){
    if(strcmp(str2,"sda1")==0){
        tmp=atof(str1)/(1024*1024);
        info.sizeHD=tmp;
        printf("info.sizeHD %.fGB\n",info.sizeHD);
    }
}
```

Figure 6 : Récupération et d'effectuation de la taille de disque dur du client



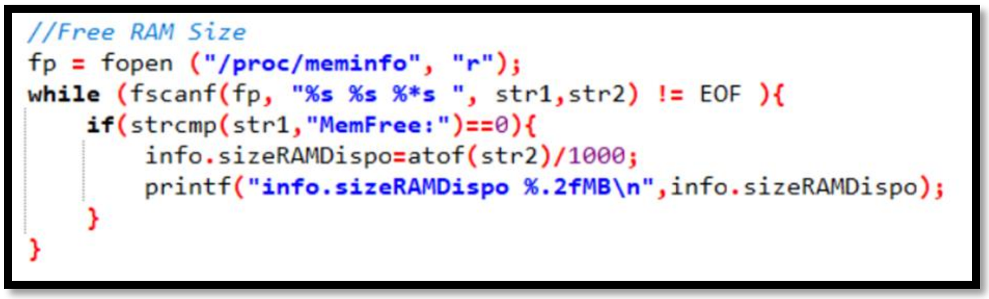
```
//Free HDD
if (statvfs("/", &stat) == 0) {
    // error happens, just quits here
    tmp = (stat.f_frsize * stat.f_bavail)/(1024*1024);
    info.sizeHDDispo=tmp;
    printf("info.sizeHDDispo %.fMB\n", info.sizeHDDispo);
```

Figure 7 : Récupération de taille de l'espace disponible du client



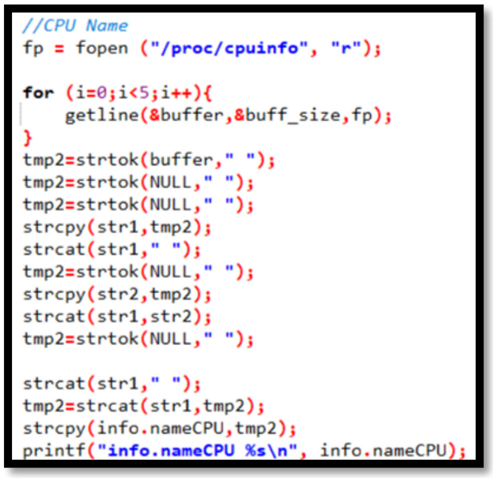
```
char *buffer, *tmp2;
size_t buff_size = 0;
//RAM Size
fp = fopen ("/proc/meminfo", "r");
while (fscanf(fp, "%s %s %*s ", str1,str2) != EOF ){
    if(strcmp(str1,"MemTotal:")==0){
        tmp=atof(str2)/1000;
        info.sizeRAM=tmp;
        printf("info.sizeRAM %.2fMB\n", info.sizeRAM);
    }
}
```

Figure 8 : Récupération de la taille de la mémoire vie du client et l'affection sur la structure



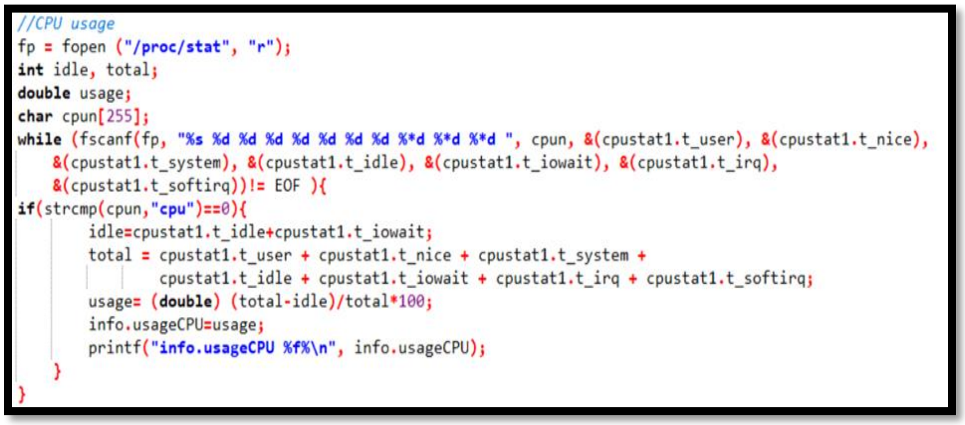
```
//Free RAM Size
fp = fopen ("/proc/meminfo", "r");
while (fscanf(fp, "%s %s %*s ", str1,str2) != EOF ){
    if(strcmp(str1,"MemFree:")==0){
        info.sizeRAMDispo=atof(str2)/1000;
        printf("info.sizeRAMDispo %.2fMB\n",info.sizeRAMDispo);
    }
}
```

Figure 9 : Récupération de la quantité de la mémoire vive disponible du client et l'affection sur la structure



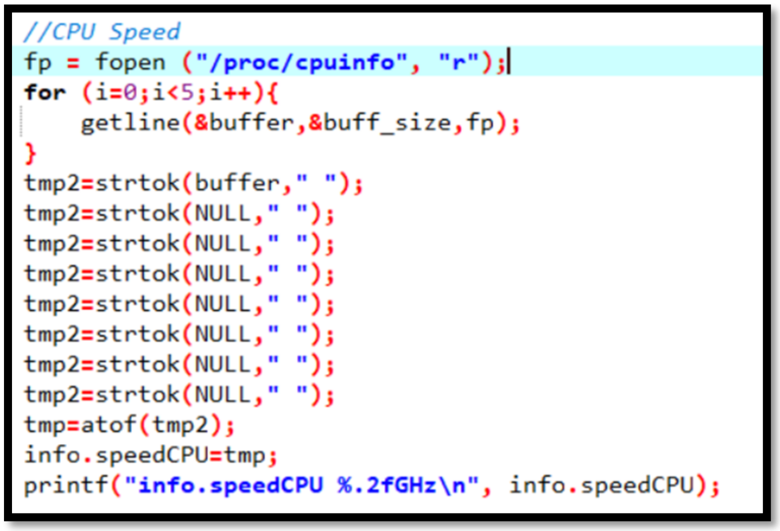
```c
//CPU Name
fp = fopen ("/proc/cpuinfo", "r");

for (i=0;i<5;i++){
    getline(&buffer,&buff_size,fp);
}
tmp2=strtok(buffer," ");
tmp2=strtok(NULL," ");
tmp2=strtok(NULL," ");
strcpy(str1,tmp2);
strcat(str1," ");
tmp2=strtok(NULL," ");
strcpy(str2,tmp2);
strcat(str1,str2);
tmp2=strtok(NULL," ");

strcat(str1," ");
tmp2=strcat(str1,tmp2);
strcpy(info.nameCPU,tmp2);
printf("info.nameCPU %s\n", info.nameCPU);
```

Figure 10 : Récupération du nom du processus du client et l'affection sur la structure



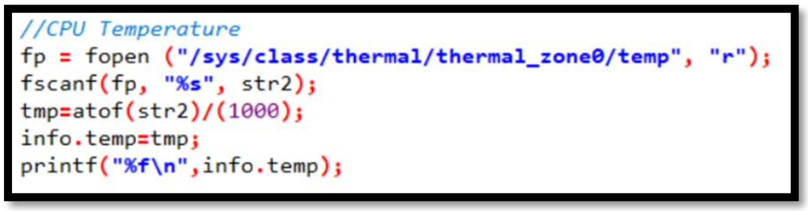
```c
//CPU usage
fp = fopen ("/proc/stat", "r");
int idle, total;
double usage;
char cpun[255];
while (fscanf(fp, "%s %d %d %d %d %d %d %d %*d %*d %*d ", cpun, &(cpustat1.t_user), &(cpustat1.t_nice),
    &(cpustat1.t_system), &(cpustat1.t_idle), &(cpustat1.t_iowait), &(cpustat1.t_irq),
    &(cpustat1.t_softirq))!= EOF ){
if(strcmp(cpun,"cpu")==0){
        idle=cpustat1.t_idle+cpustat1.t_iowait;
        total = cpustat1.t_user + cpustat1.t_nice + cpustat1.t_system +
                cpustat1.t_idle + cpustat1.t_iowait + cpustat1.t_irq + cpustat1.t_softirq;
        usage= (double) (total-idle)/total*100;
        info.usageCPU=usage;
        printf("info.usageCPU %f%\n", info.usageCPU);
    }
}
```

Figure 11 : Récupération de la fréquence du processus du client et l'affection sur la structure



```
//CPU Speed
fp = fopen ("/proc/cpuinfo", "r");
for (i=0;i<5;i++){
    getline(&buffer,&buff_size,fp);
}
tmp2=strtok(buffer," ");
tmp2=strtok(NULL," ");
tmp2=strtok(NULL," ");
tmp2=strtok(NULL," ");
tmp2=strtok(NULL," ");
tmp2=strtok(NULL," ");
tmp2=strtok(NULL," ");
tmp2=strtok(NULL," ");
tmp=atof(tmp2);
info.speedCPU=tmp;
printf("info.speedCPU %.2fGHz\n", info.speedCPU);
```

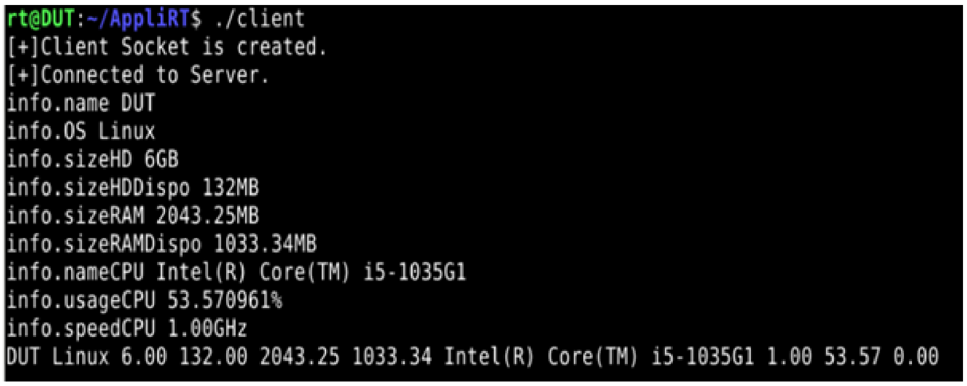
Figure 12 : Récupération de la charge d'utilisation du processus du client et l'affection sur la structure



```
//CPU Temperature
fp = fopen ("/sys/class/thermal/thermal_zone0/temp", "r");
fscanf(fp, "%s", str2);
tmp=atof(str2)/(1000);
info.temp=tmp;
printf("%f\n",info.temp);
```

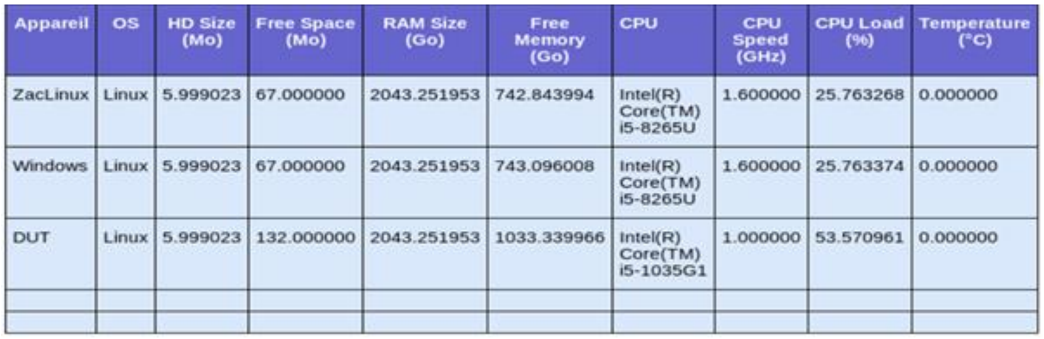
Figure 13 : Récupération de la température du processus du client et l'affection sur la structure

c. L'exemple de message envoyé par client :



```
rt@DUT:~/AppliRT$ ./client
[+]Client Socket is created.
[+]Connected to Server.
info.name DUT
info.OS Linux
info.sizeHD 6GB
info.sizeHDDispo 132MB
info.sizeRAM 2043.25MB
info.sizeRAMDispo 1033.34MB
info.nameCPU Intel(R) Core(TM) i5-1035G1
info.usageCPU 53.570961%
info.speedCPU 1.00GHz
DUT Linux 6.00 132.00 2043.25 1033.34 Intel(R) Core(TM) i5-1035G1 1.00 53.57 0.00
```

3. Le serveur met à jour les données sur **l'interface web** chaque 10s.
    - ➢ Le serveur va chercher dans la page html la ligne concernant ce client
    - ➢ Il va remplacer les informations si le client déjà existe, sinon il va ajouter les infos concernées.



Appareil	OS	HD Size (Mo)	Free Space (Mo)	RAM Size (Go)	Free Memory (Go)	CPU	CPU Speed (GHz)	CPU Load (%)	Temperature (°C)
ZacLinux	Linux	5.999023	67.000000	2043.251953	742.843994	Intel(R) Core(TM) i5-8265U	1.600000	25.763268	0.000000
Windows	Linux	5.999023	67.000000	2043.251953	743.096008	Intel(R) Core(TM) i5-8265U	1.600000	25.763374	0.000000
DUT	Linux	5.999023	132.000000	2043.251953	1033.339966	Intel(R) Core(TM) i5-1035G1	1.000000	53.570961	0.000000

❖ Les problèmes rencontrés

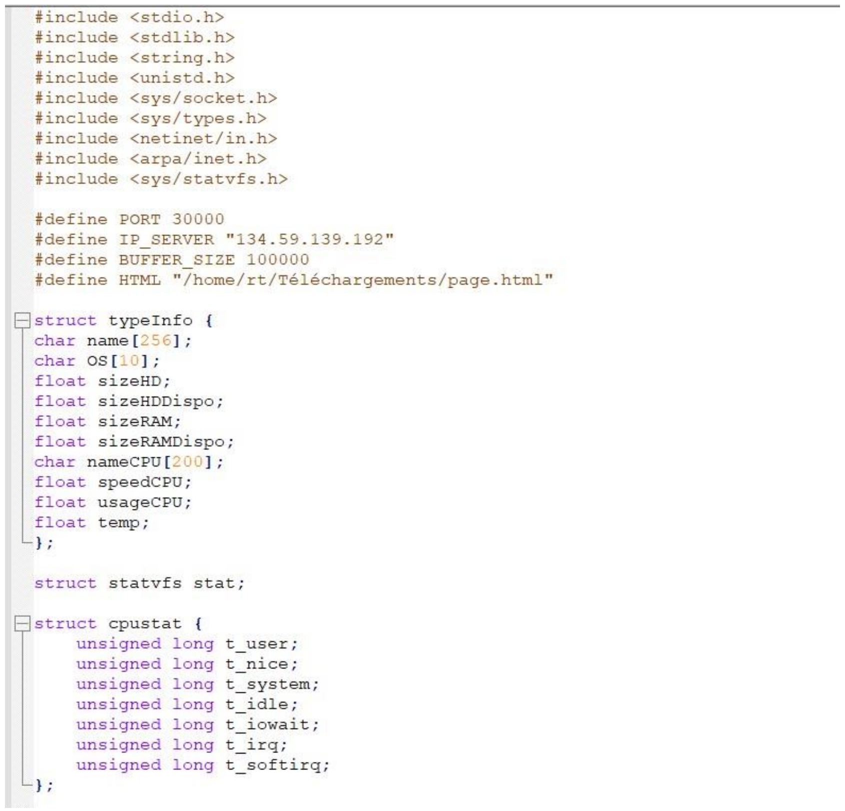
- ➢ En général, l'utilisation de langage C était difficile et compliqué pour nous comme c'est un langage très explicite, et étant donné que nous avons vu seulement la base de ce langage en S1. Nous avons dû revisiter le cours et se renseigner sur Internet.
- ➢ Nous rencontrons aussi des difficultés à comprendre les notions de socket. Donc nous nous sommes renseignés encore une fois sur Internet et lire le cours de réseaux pour comprendre le principe
- ➢ Difficultés de "parser" les lignes d'un fichier pour extraire les informations recherchées. Nous avons plus d'expérience d'éffectuer le traitement des lignes en Python avec le module Regex par exemple. En C, nous avons dû faire appels à des différents fonctions comme strtok(), getline() etc.
- ➢ Pas d'affection sur les Strings et les problèmes de déclaration. Nous avons eu beaucoup d'erreur dans le programme à cause de type variable incompatible.
- ➢ Problème d'affichage des données sur l'interface Web en temps réel. Nous avons donc décidé de modifier le fichier html de manière constante (chaque 10s) avec les méthodes comme fopen, fputs etc.

❖ Améliorations possibles

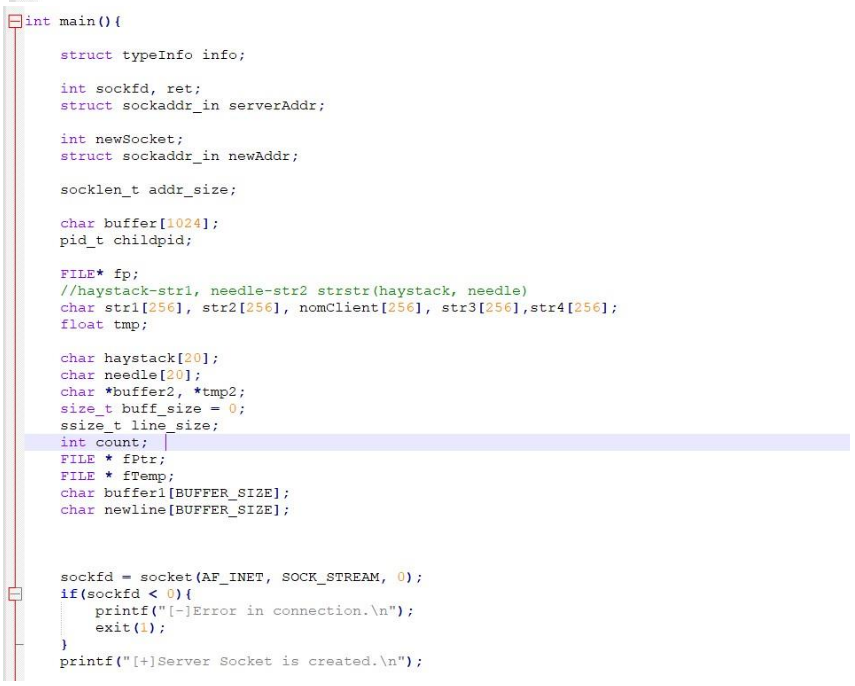
- ➢ Créer des fonctions dans un fichier et les appeler dans code source server.c et client.c pour simplifier les codes. Par exemple, les fonctions pour récupérer et traiter les informations sur la machine client
- ➢ Afficher les adresses IP de machine client sur l'interface web
- ➢ Ajouter la possibilité d'éteindre ou d'allumer une machine supervisée en ajoutant les lignes comme :
    - o System(/sbin/shutdown) - pour éteindre
    - o System(/sbin/reboot) - pour redémarrer
- ➢ Héberger l'interface web sur Internet en utilisant un serveur Apache et CGI. Dynamiser le fichier HTML (pour qu'il puisse ajouter les lignes automatiquement dès qu'il y a une nouvelle machine qui sera connecté au serveur).

Les codes sources

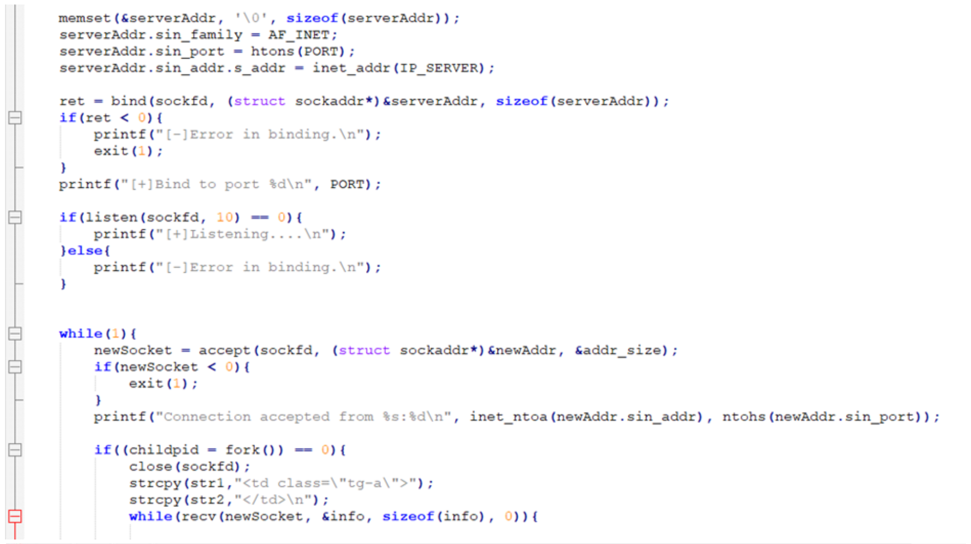
1. Codes sources server.c



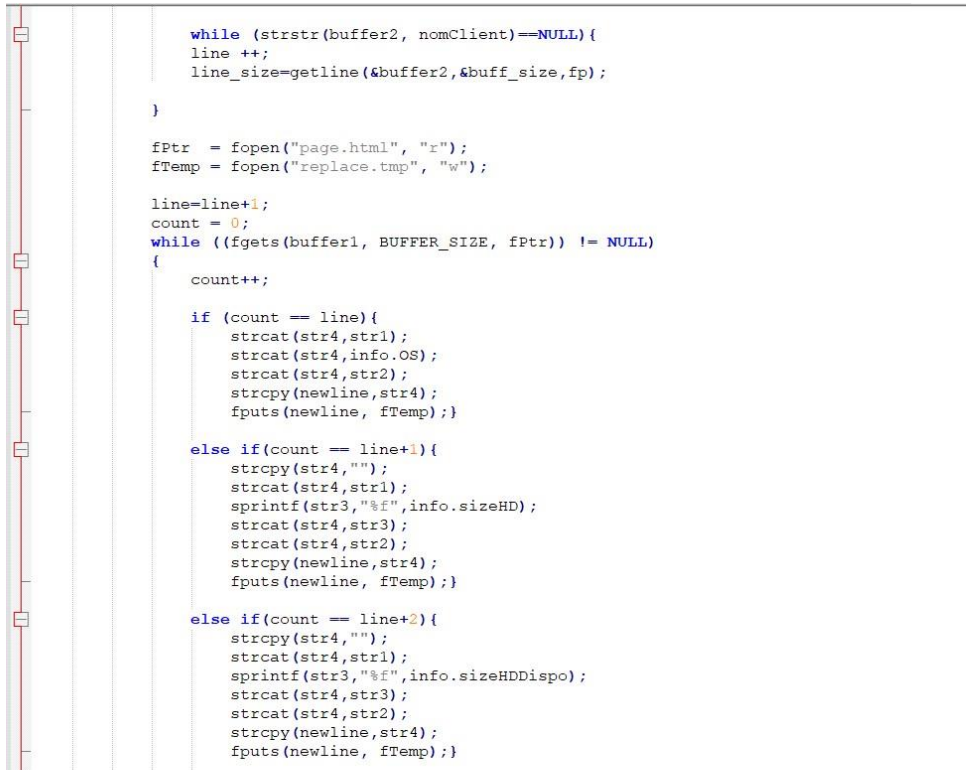
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/statvfs.h>

#define PORT 30000
#define IP_SERVER "134.59.139.192"
#define BUFFER_SIZE 100000
#define HTML "/home/rt/Téléchargements/page.html"

struct typeInfo {
char name[256];
char OS[10];
float sizeHD;
float sizeHDDispo;
float sizeRAM;
float sizeRAMDispo;
char nameCPU[200];
float speedCPU;
float usageCPU;
float temp;
};

struct statvfs stat;

struct cpustat {
    unsigned long t_user;
    unsigned long t_nice;
    unsigned long t_system;
    unsigned long t_idle;
    unsigned long t_iowait;
    unsigned long t_irq;
    unsigned long t_softirq;
};
```



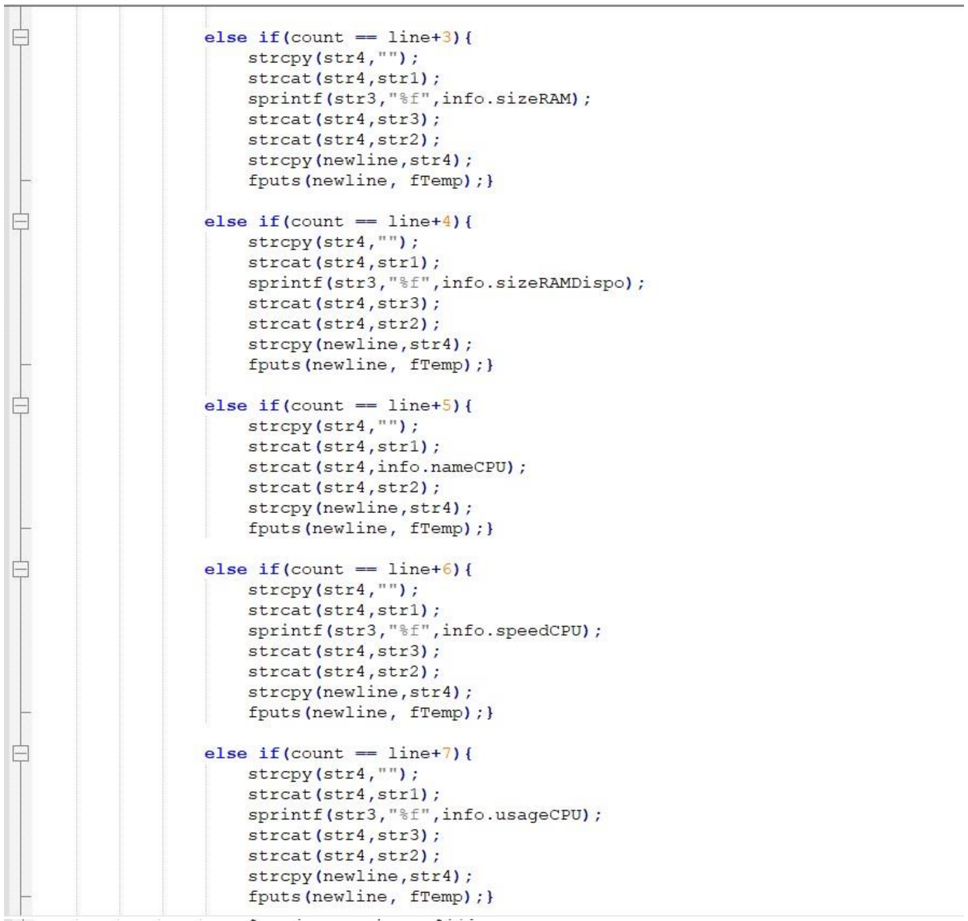
```c
int main(){

    struct typeInfo info;

    int sockfd, ret;
    struct sockaddr_in serverAddr;

    int newSocket;
    struct sockaddr_in newAddr;

    socklen_t addr_size;

    char buffer[1024];
    pid_t childpid;

    FILE* fp;
    //haystack-str1, needle-str2 strstr(haystack, needle)
    char str1[256], str2[256], nomClient[256], str3[256],str4[256];
    float tmp;

    char haystack[20];
    char needle[20];
    char *buffer2, *tmp2;
    size_t buff_size = 0;
    ssize_t line_size;
    int count;
    FILE * fPtr;
    FILE * fTemp;
    char buffer1[BUFFER_SIZE];
    char newline[BUFFER_SIZE];



    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if(sockfd < 0){
        printf("[-]Error in connection.\n");
        exit(1);
    }
    printf("[+]Server Socket is created.\n");
```



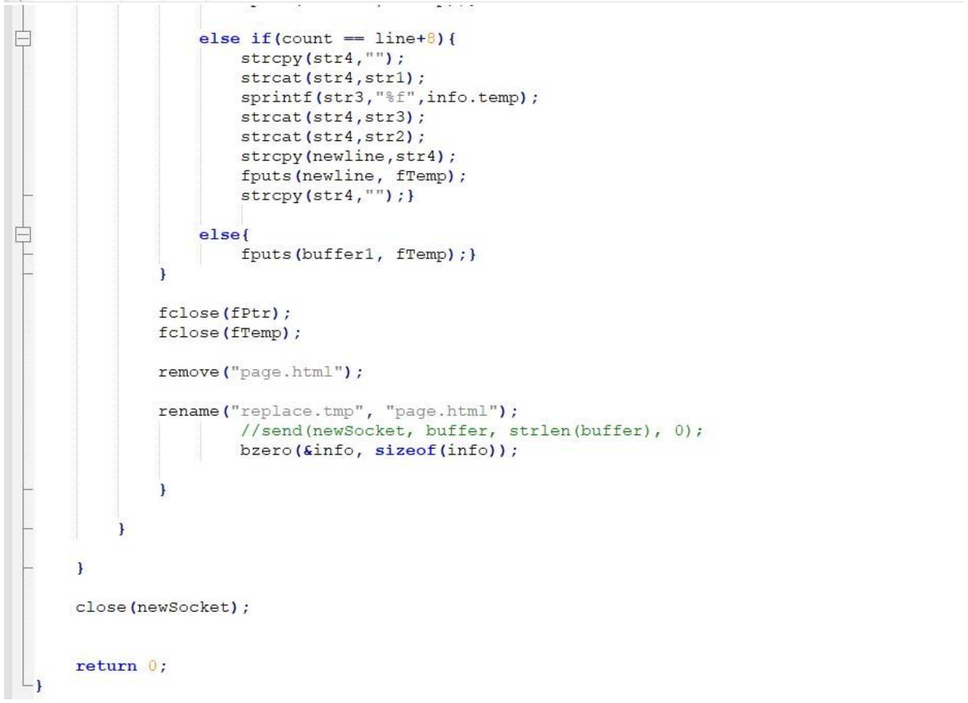
```
memset(&serverAddr, '\0', sizeof(serverAddr));
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(PORT);
serverAddr.sin_addr.s_addr = inet_addr(IP_SERVER);

ret = bind(sockfd, (struct sockaddr*)&serverAddr, sizeof(serverAddr));
if(ret < 0){
    printf("[-]Error in binding.\n");
    exit(1);
}
printf("[+]Bind to port %d\n", PORT);

if(listen(sockfd, 10) == 0){
    printf("[+]Listening....\n");
}else{
    printf("[-]Error in binding.\n");
}


while(1){
    newSocket = accept(sockfd, (struct sockaddr*)&newAddr, &addr_size);
    if(newSocket < 0){
        exit(1);
    }
    printf("Connection accepted from %s:%d\n", inet_ntoa(newAddr.sin_addr), ntohs(newAddr.sin_port));

    if((childpid = fork()) == 0){
        close(sockfd);
        strcpy(str1,"<td class=\"tg-a\">");
        strcpy(str2,"</td>\n");
        while(recv(newSocket, &info, sizeof(info), 0)){
```



```
    while (strstr(buffer2, nomClient)==NULL){
    line ++;
    line_size=getline(&buffer2,&buff_size,fp);

}

fPtr  = fopen("page.html", "r");
fTemp = fopen("replace.tmp", "w");

line=line+1;
count = 0;
while ((fgets(buffer1, BUFFER_SIZE, fPtr)) != NULL)
{
    count++;

    if (count == line){
        strcat(str4,str1);
        strcat(str4,info.OS);
        strcat(str4,str2);
        strcpy(newline,str4);
        fputs(newline, fTemp);}

    else if(count == line+1){
        strcpy(str4,"");
        strcat(str4,str1);
        sprintf(str3,"%f",info.sizeHD);
        strcat(str4,str3);
        strcat(str4,str2);
        strcpy(newline,str4);
        fputs(newline, fTemp);}

    else if(count == line+2){
        strcpy(str4,"");
        strcat(str4,str1);
        sprintf(str3,"%f",info.sizeHDDispo);
        strcat(str4,str3);
        strcat(str4,str2);
        strcpy(newline,str4);
        fputs(newline, fTemp);}
```

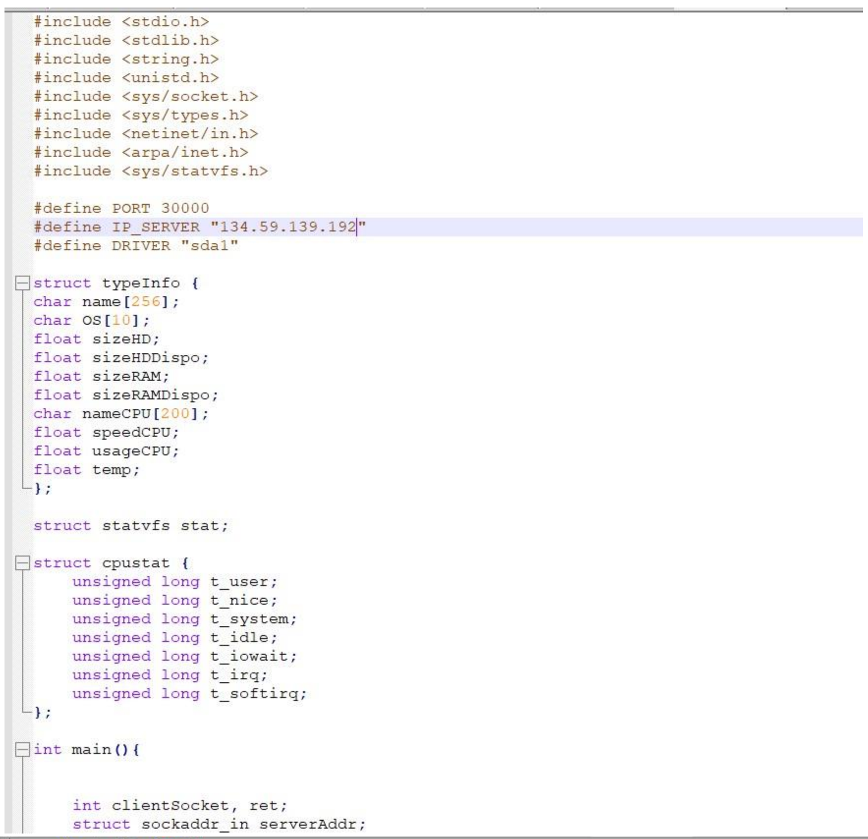


```c
                else if(count == line+3){
                    strcpy(str4,"");
                    strcat(str4,str1);
                    sprintf(str3,"%f",info.sizeRAM);
                    strcat(str4,str3);
                    strcat(str4,str2);
                    strcpy(newline,str4);
                    fputs(newline, fTemp);}

                else if(count == line+4){
                    strcpy(str4,"");
                    strcat(str4,str1);
                    sprintf(str3,"%f",info.sizeRAMDispo);
                    strcat(str4,str3);
                    strcat(str4,str2);
                    strcpy(newline,str4);
                    fputs(newline, fTemp);}

                else if(count == line+5){
                    strcpy(str4,"");
                    strcat(str4,str1);
                    strcat(str4,info.nameCPU);
                    strcat(str4,str2);
                    strcpy(newline,str4);
                    fputs(newline, fTemp);}

                else if(count == line+6){
                    strcpy(str4,"");
                    strcat(str4,str1);
                    sprintf(str3,"%f",info.speedCPU);
                    strcat(str4,str3);
                    strcat(str4,str2);
                    strcpy(newline,str4);
                    fputs(newline, fTemp);}

                else if(count == line+7){
                    strcpy(str4,"");
                    strcat(str4,str1);
                    sprintf(str3,"%f",info.usageCPU);
                    strcat(str4,str3);
                    strcat(str4,str2);
                    strcpy(newline,str4);
                    fputs(newline, fTemp);}
```

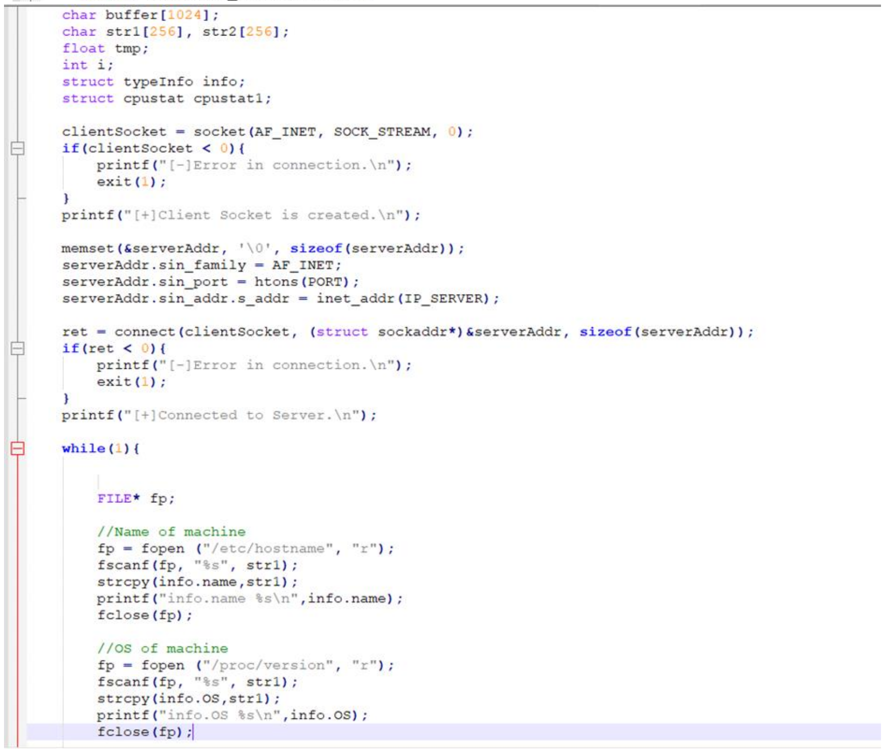


```c
                else if(count == line+8){
                    strcpy(str4,"");
                    strcat(str4,str1);
                    sprintf(str3,"%f",info.temp);
                    strcat(str4,str3);
                    strcat(str4,str2);
                    strcpy(newline,str4);
                    fputs(newline, fTemp);
                    strcpy(str4,"");}

                else{
                    fputs(buffer1, fTemp);}
            }

            fclose(fPtr);
            fclose(fTemp);

            remove("page.html");

            rename("replace.tmp", "page.html");
                //send(newSocket, buffer, strlen(buffer), 0);
                bzero(&info, sizeof(info));

            }

        }

    }

    close(newSocket);


    return 0;
}
```

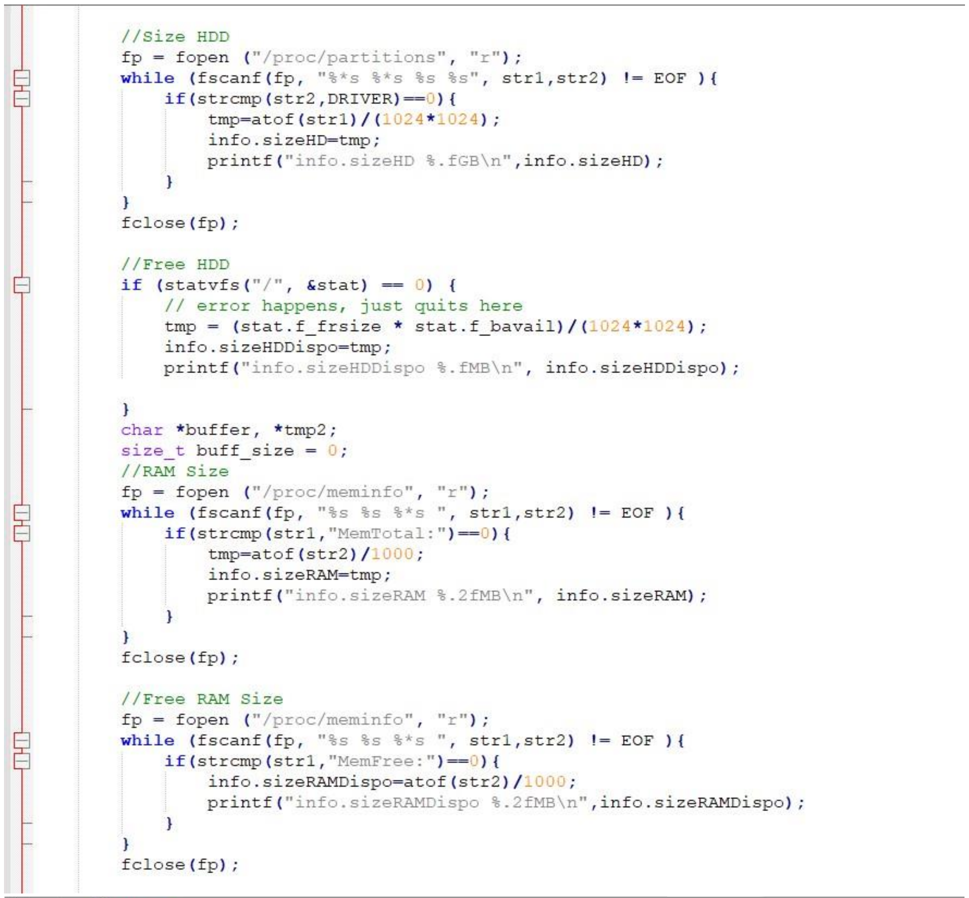
2. Codes sources client.c (Linux)



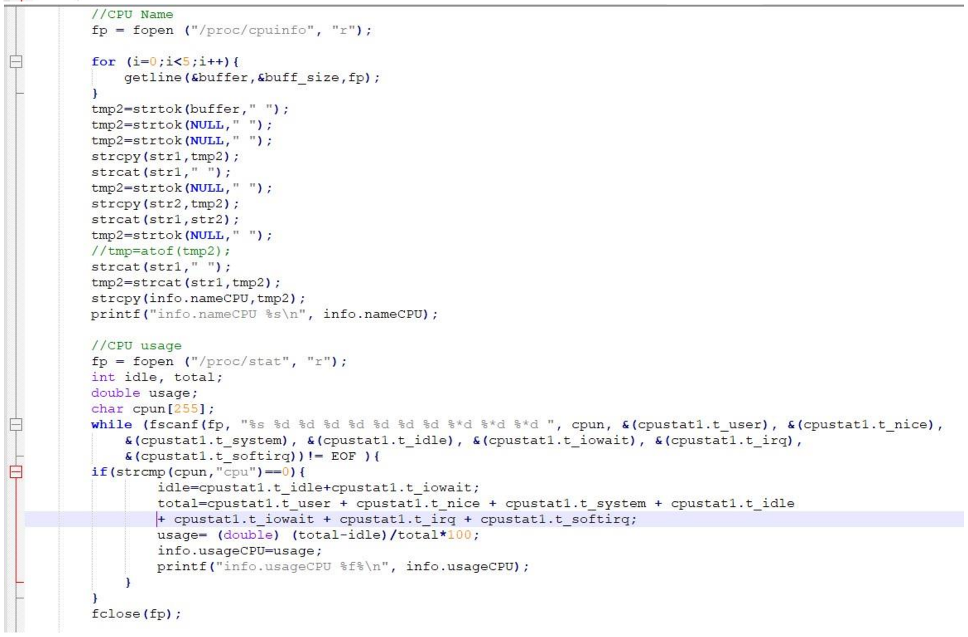
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/statvfs.h>

#define PORT 30000
#define IP_SERVER "134.59.139.192"
#define DRIVER "sda1"

struct typeInfo {
char name[256];
char OS[10];
float sizeHD;
float sizeHDDispo;
float sizeRAM;
float sizeRAMDispo;
char nameCPU[200];
float speedCPU;
float usageCPU;
float temp;
};

struct statvfs stat;

struct cpustat {
    unsigned long t_user;
    unsigned long t_nice;
    unsigned long t_system;
    unsigned long t_idle;
    unsigned long t_iowait;
    unsigned long t_irq;
    unsigned long t_softirq;
};

int main(){


    int clientSocket, ret;
    struct sockaddr_in serverAddr;
```



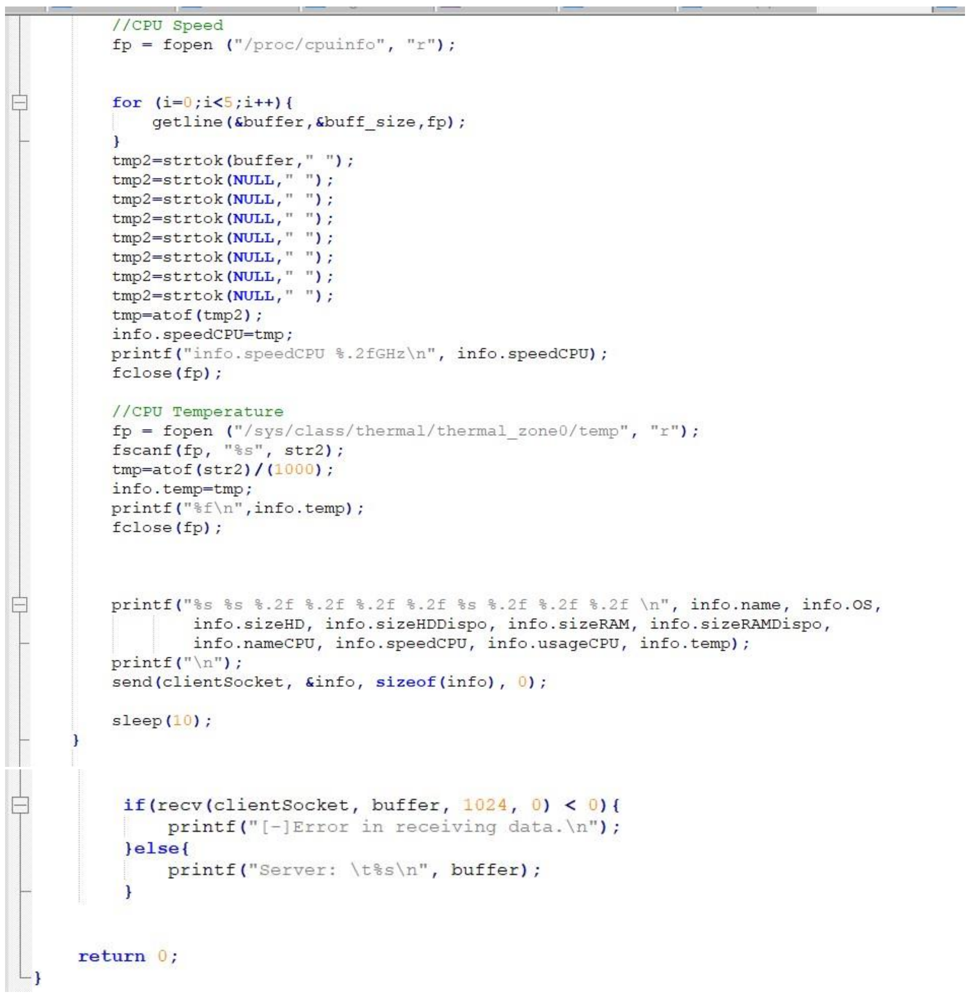
```c
    char buffer[1024];
    char str1[256], str2[256];
    float tmp;
    int i;
    struct typeInfo info;
    struct cpustat cpustat1;

    clientSocket = socket(AF_INET, SOCK_STREAM, 0);
    if(clientSocket < 0){
        printf("[-]Error in connection.\n");
        exit(1);
    }
    printf("[+]Client Socket is created.\n");

    memset(&serverAddr, '\0', sizeof(serverAddr));
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(PORT);
    serverAddr.sin_addr.s_addr = inet_addr(IP_SERVER);

    ret = connect(clientSocket, (struct sockaddr*)&serverAddr, sizeof(serverAddr));
    if(ret < 0){
        printf("[-]Error in connection.\n");
        exit(1);
    }
    printf("[+]Connected to Server.\n");

    while(1){


        FILE* fp;

        //Name of machine
        fp = fopen ("/etc/hostname", "r");
        fscanf(fp, "%s", str1);
        strcpy(info.name,str1);
        printf("info.name %s\n",info.name);
        fclose(fp);

        //OS of machine
        fp = fopen ("/proc/version", "r");
        fscanf(fp, "%s", str1);
        strcpy(info.OS,str1);
        printf("info.OS %s\n",info.OS);
        fclose(fp);
```



```c
//Size HDD
fp = fopen ("/proc/partitions", "r");
while (fscanf(fp, "%*s %*s %s %s", str1,str2) != EOF ){
    if(strcmp(str2,DRIVER)==0){
        tmp=atof(str1)/(1024*1024);
        info.sizeHD=tmp;
        printf("info.sizeHD %.fGB\n",info.sizeHD);
    }
}
fclose(fp);

//Free HDD
if (statvfs("/", &stat) == 0) {
    // error happens, just quits here
    tmp = (stat.f_frsize * stat.f_bavail)/(1024*1024);
    info.sizeHDDispo=tmp;
    printf("info.sizeHDDispo %.fMB\n", info.sizeHDDispo);

}
char *buffer, *tmp2;
size_t buff_size = 0;
//RAM Size
fp = fopen ("/proc/meminfo", "r");
while (fscanf(fp, "%s %s %*s ", str1,str2) != EOF ){
    if(strcmp(str1,"MemTotal:")==0){
        tmp=atof(str2)/1000;
        info.sizeRAM=tmp;
        printf("info.sizeRAM %.2fMB\n", info.sizeRAM);
    }
}
fclose(fp);

//Free RAM Size
fp = fopen ("/proc/meminfo", "r");
while (fscanf(fp, "%s %s %*s ", str1,str2) != EOF ){
    if(strcmp(str1,"MemFree:")==0){
        info.sizeRAMDispo=atof(str2)/1000;
        printf("info.sizeRAMDispo %.2fMB\n",info.sizeRAMDispo);
    }
}
fclose(fp);
```

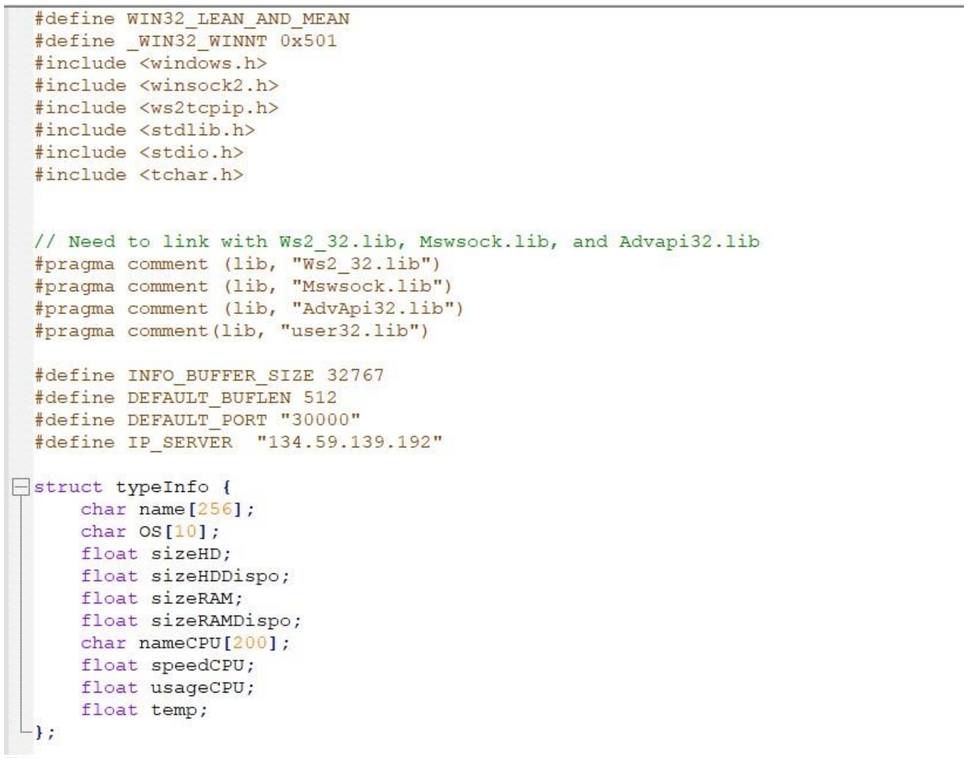


```c
//CPU Name
fp = fopen ("/proc/cpuinfo", "r");

for (i=0;i<5;i++){
    getline(&buffer,&buff_size,fp);
}
tmp2=strtok(buffer," ");
tmp2=strtok(NULL," ");
tmp2=strtok(NULL," ");
strcpy(str1,tmp2);
strcat(str1," ");
tmp2=strtok(NULL," ");
strcpy(str2,tmp2);
strcat(str1,str2);
tmp2=strtok(NULL," ");
//tmp=atof(tmp2);
strcat(str1," ");
tmp2=strcat(str1,tmp2);
strcpy(info.nameCPU,tmp2);
printf("info.nameCPU %s\n", info.nameCPU);

//CPU usage
fp = fopen ("/proc/stat", "r");
int idle, total;
double usage;
char cpun[255];
while (fscanf(fp, "%s %d %d %d %d %d %d %d %*d %*d %*d ", cpun, &(cpustat1.t_user), &(cpustat1.t_nice),
    &(cpustat1.t_system), &(cpustat1.t_idle), &(cpustat1.t_iowait), &(cpustat1.t_irq),
    &(cpustat1.t_softirq))!= EOF ){
if(strcmp(cpun,"cpu")==0){
        idle=cpustat1.t_idle+cpustat1.t_iowait;
        total=cpustat1.t_user + cpustat1.t_nice + cpustat1.t_system + cpustat1.t_idle
        + cpustat1.t_iowait + cpustat1.t_irq + cpustat1.t_softirq;
        usage= (double) (total-idle)/total*100;
        info.usageCPU=usage;
        printf("info.usageCPU %f%\n", info.usageCPU);
    }
}
fclose(fp);
```

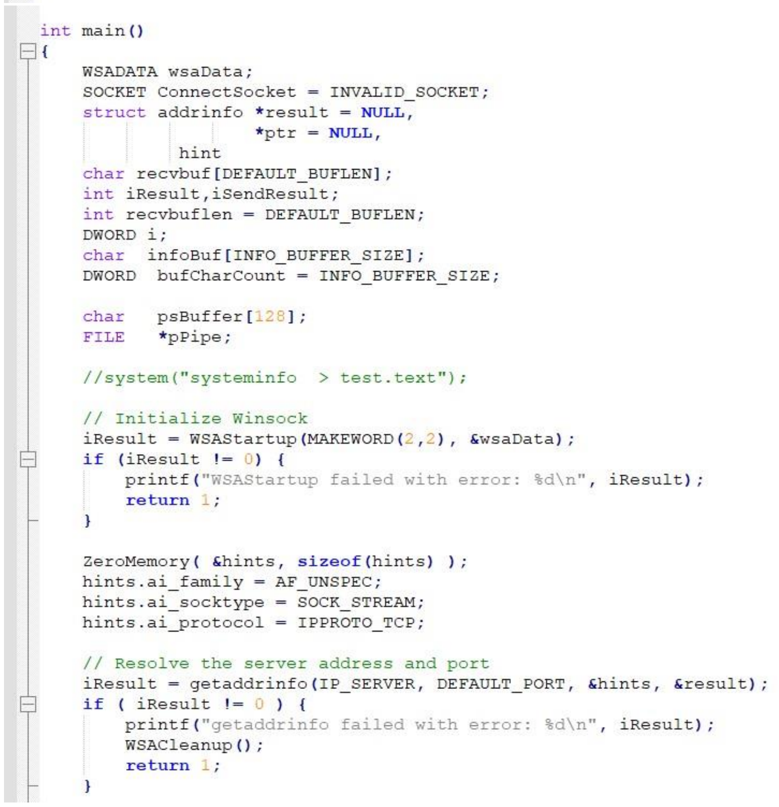


```
        //CPU Speed
        fp = fopen ("/proc/cpuinfo", "r");


        for (i=0;i<5;i++){
            getline(&buffer,&buff_size,fp);
        }
        tmp2=strtok(buffer," ");
        tmp2=strtok(NULL," ");
        tmp2=strtok(NULL," ");
        tmp2=strtok(NULL," ");
        tmp2=strtok(NULL," ");
        tmp2=strtok(NULL," ");
        tmp2=strtok(NULL," ");
        tmp2=strtok(NULL," ");
        tmp=atof(tmp2);
        info.speedCPU=tmp;
        printf("info.speedCPU %.2fGHz\n", info.speedCPU);
        fclose(fp);

        //CPU Temperature
        fp = fopen ("/sys/class/thermal/thermal_zone0/temp", "r");
        fscanf(fp, "%s", str2);
        tmp=atof(str2)/(1000);
        info.temp=tmp;
        printf("%f\n",info.temp);
        fclose(fp);



        printf("%s %s %.2f %.2f %.2f %.2f %s %.2f %.2f %.2f \n", info.name, info.OS,
                info.sizeHD, info.sizeHDDispo, info.sizeRAM, info.sizeRAMDispo,
                info.nameCPU, info.speedCPU, info.usageCPU, info.temp);
        printf("\n");
        send(clientSocket, &info, sizeof(info), 0);

        sleep(10);
    }


     if(recv(clientSocket, buffer, 1024, 0) < 0){
         printf("[-]Error in receiving data.\n");
     }else{
         printf("Server: \t%s\n", buffer);
     }


    return 0;
}
```

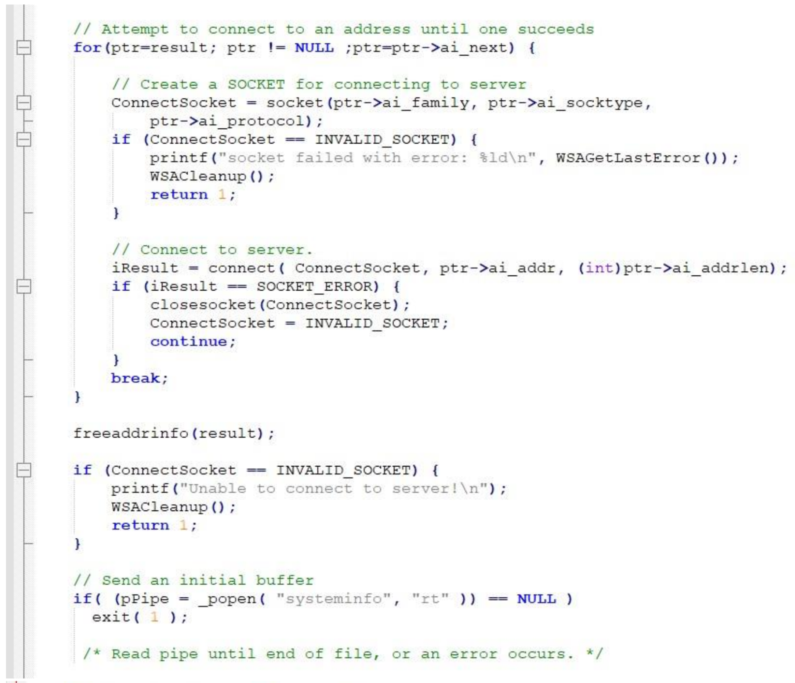
3. Codes sources clientW.c (Windows)



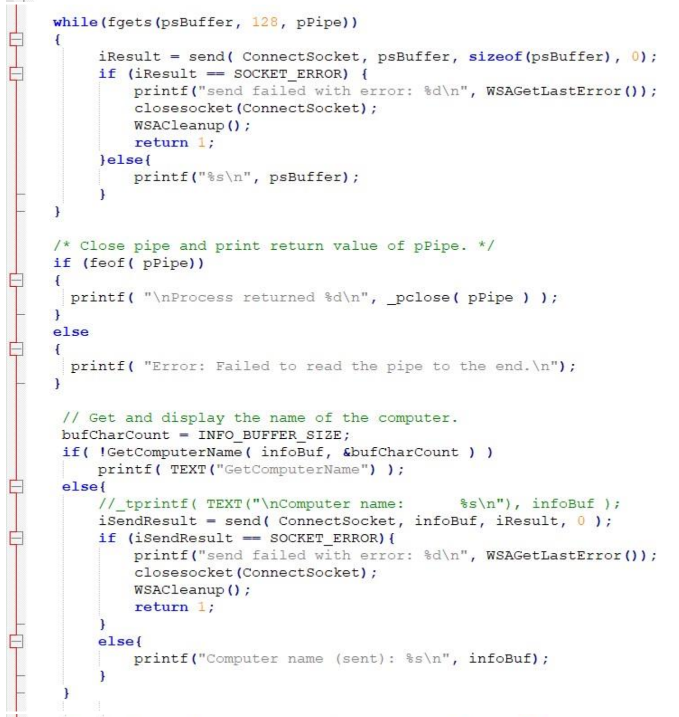
```c
#define WIN32_LEAN_AND_MEAN
#define _WIN32_WINNT 0x501
#include <windows.h>
#include <winsock2.h>
#include <ws2tcpip.h>
#include <stdlib.h>
#include <stdio.h>
#include <tchar.h>


// Need to link with Ws2_32.lib, Mswsock.lib, and Advapi32.lib
#pragma comment (lib, "Ws2_32.lib")
#pragma comment (lib, "Mswsock.lib")
#pragma comment (lib, "AdvApi32.lib")
#pragma comment(lib, "user32.lib")

#define INFO_BUFFER_SIZE 32767
#define DEFAULT_BUFLEN 512
#define DEFAULT_PORT "30000"
#define IP_SERVER  "134.59.139.192"

struct typeInfo {
    char name[256];
    char OS[10];
    float sizeHD;
    float sizeHDDispo;
    float sizeRAM;
    float sizeRAMDispo;
    char nameCPU[200];
    float speedCPU;
    float usageCPU;
    float temp;
};
```



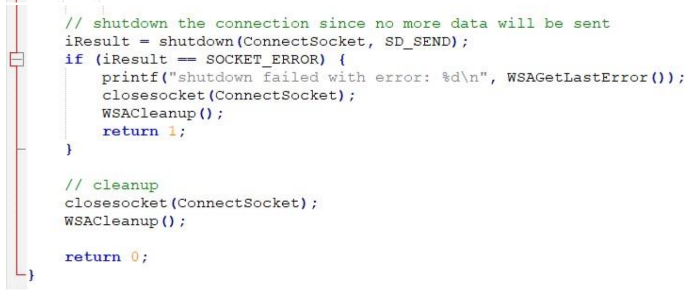
```c
int main()
{
    WSADATA wsaData;
    SOCKET ConnectSocket = INVALID_SOCKET;
    struct addrinfo *result = NULL,
                    *ptr = NULL,
                hint
    char recvbuf[DEFAULT_BUFLEN];
    int iResult,iSendResult;
    int recvbuflen = DEFAULT_BUFLEN;
    DWORD i;
    char  infoBuf[INFO_BUFFER_SIZE];
    DWORD  bufCharCount = INFO_BUFFER_SIZE;

    char   psBuffer[128];
    FILE   *pPipe;

    //system("systeminfo  > test.text");

    // Initialize Winsock
    iResult = WSAStartup(MAKEWORD(2,2), &wsaData);
    if (iResult != 0) {
        printf("WSAStartup failed with error: %d\n", iResult);
        return 1;
    }

    ZeroMemory( &hints, sizeof(hints) );
    hints.ai_family = AF_UNSPEC;
    hints.ai_socktype = SOCK_STREAM;
    hints.ai_protocol = IPPROTO_TCP;

    // Resolve the server address and port
    iResult = getaddrinfo(IP_SERVER, DEFAULT_PORT, &hints, &result);
    if ( iResult != 0 ) {
        printf("getaddrinfo failed with error: %d\n", iResult);
        WSACleanup();
        return 1;
    }
```



```
    // Attempt to connect to an address until one succeeds
    for(ptr=result; ptr != NULL ;ptr=ptr->ai_next) {

        // Create a SOCKET for connecting to server
        ConnectSocket = socket(ptr->ai_family, ptr->ai_socktype,
            ptr->ai_protocol);
        if (ConnectSocket == INVALID_SOCKET) {
            printf("socket failed with error: %ld\n", WSAGetLastError());
            WSACleanup();
            return 1;
        }

        // Connect to server.
        iResult = connect( ConnectSocket, ptr->ai_addr, (int)ptr->ai_addrlen);
        if (iResult == SOCKET_ERROR) {
            closesocket(ConnectSocket);
            ConnectSocket = INVALID_SOCKET;
            continue;
        }
        break;
    }

    freeaddrinfo(result);

    if (ConnectSocket == INVALID_SOCKET) {
        printf("Unable to connect to server!\n");
        WSACleanup();
        return 1;
    }

    // Send an initial buffer
    if( (pPipe = _popen( "systeminfo", "rt" )) == NULL )
      exit( 1 );

     /* Read pipe until end of file, or an error occurs. */
```

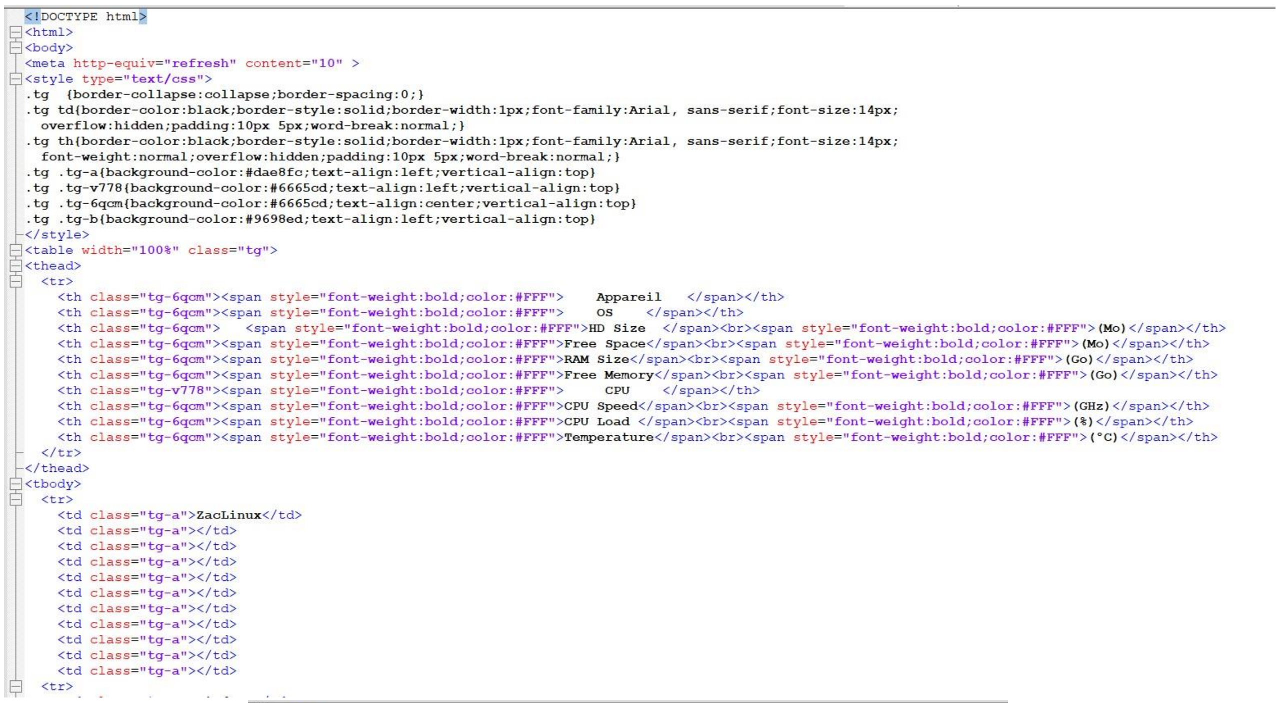


```
   while(fgets(psBuffer, 128, pPipe))
   {
       iResult = send( ConnectSocket, psBuffer, sizeof(psBuffer), 0);
       if (iResult == SOCKET_ERROR) {
           printf("send failed with error: %d\n", WSAGetLastError());
           closesocket(ConnectSocket);
           WSACleanup();
           return 1;
       }else{
           printf("%s\n", psBuffer);
       }
   }

   /* Close pipe and print return value of pPipe. */
   if (feof( pPipe))
   {
     printf( "\nProcess returned %d\n", _pclose( pPipe ) );
   }
   else
   {
     printf( "Error: Failed to read the pipe to the end.\n");
   }

    // Get and display the name of the computer.
    bufCharCount = INFO_BUFFER_SIZE;
    if( !GetComputerName( infoBuf, &bufCharCount ) )
        printf( TEXT("GetComputerName") );
    else{
        //_tprintf( TEXT("\nComputer name:      %s\n"), infoBuf );
        iSendResult = send( ConnectSocket, infoBuf, iResult, 0 );
        if (iSendResult == SOCKET_ERROR){
            printf("send failed with error: %d\n", WSAGetLastError());
            closesocket(ConnectSocket);
            WSACleanup();
            return 1;
        }
        else{
            printf("Computer name (sent): %s\n", infoBuf);
        }
    }
```

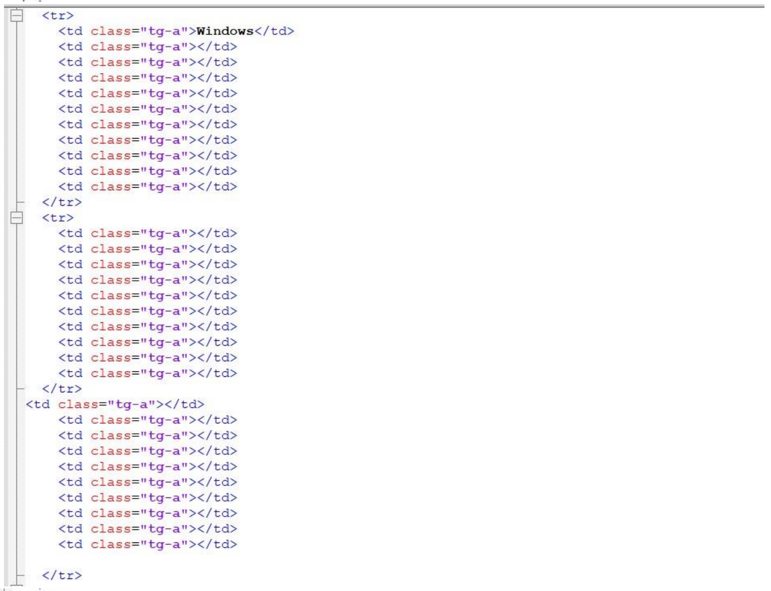


```
    // shutdown the connection since no more data will be sent
    iResult = shutdown(ConnectSocket, SD_SEND);
    if (iResult == SOCKET_ERROR) {
        printf("shutdown failed with error: %d\n", WSAGetLastError());
        closesocket(ConnectSocket);
        WSACleanup();
        return 1;
    }

    // cleanup
    closesocket(ConnectSocket);
    WSACleanup();

    return 0;
}
```

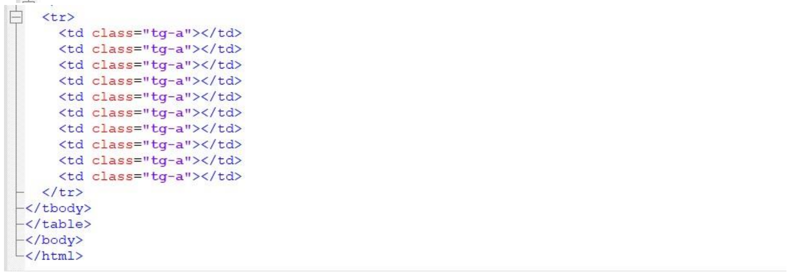
4. Codes HTML page.html



```html
<!DOCTYPE html>
<html>
<body>
<meta http-equiv="refresh" content="10" >
<style type="text/css">
.tg  {border-collapse:collapse;border-spacing:0;}
.tg td{border-color:black;border-style:solid;border-width:1px;font-family:Arial, sans-serif;font-size:14px;
  overflow:hidden;padding:10px 5px;word-break:normal;}
.tg th{border-color:black;border-style:solid;border-width:1px;font-family:Arial, sans-serif;font-size:14px;
  font-weight:normal;overflow:hidden;padding:10px 5px;word-break:normal;}
.tg .tg-a{background-color:#dae8fc;text-align:left;vertical-align:top}
.tg .tg-v778{background-color:#6665cd;text-align:left;vertical-align:top}
.tg .tg-6qcm{background-color:#6665cd;text-align:center;vertical-align:top}
.tg .tg-b{background-color:#9698ed;text-align:left;vertical-align:top}
</style>
<table width="100%" class="tg">
<thead>
  <tr>
    <th class="tg-6qcm"><span style="font-weight:bold;color:#FFF">    Appareil   </span></th>
    <th class="tg-6qcm"><span style="font-weight:bold;color:#FFF">    OS    </span></th>
    <th class="tg-6qcm">   <span style="font-weight:bold;color:#FFF">HD Size  </span><br><span style="font-weight:bold;color:#FFF">(Mo)</span></th>
    <th class="tg-6qcm"><span style="font-weight:bold;color:#FFF">Free Space</span><br><span style="font-weight:bold;color:#FFF">(Mo)</span></th>
    <th class="tg-6qcm"><span style="font-weight:bold;color:#FFF">RAM Size</span><br><span style="font-weight:bold;color:#FFF">(Go)</span></th>
    <th class="tg-6qcm"><span style="font-weight:bold;color:#FFF">Free Memory</span><br><span style="font-weight:bold;color:#FFF">(Go)</span></th>
    <th class="tg-v778"><span style="font-weight:bold;color:#FFF">     CPU    </span></th>
    <th class="tg-6qcm"><span style="font-weight:bold;color:#FFF">CPU Speed</span><br><span style="font-weight:bold;color:#FFF">(GHz)</span></th>
    <th class="tg-6qcm"><span style="font-weight:bold;color:#FFF">CPU Load </span><br><span style="font-weight:bold;color:#FFF">(%)</span></th>
    <th class="tg-6qcm"><span style="font-weight:bold;color:#FFF">Temperature</span><br><span style="font-weight:bold;color:#FFF">(°C)</span></th>
  </tr>
</thead>
<tbody>
  <tr>
    <td class="tg-a">ZacLinux</td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
  <tr>
```



```html
  <tr>
    <td class="tg-a">Windows</td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
  </tr>
  <tr>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
  </tr>
<td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>

  </tr>
```



```html
  <tr>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
    <td class="tg-a"></td>
  </tr>
</tbody>
</table>
</body>
</html>
```