Lab3: Computer Networks

Subject

- Running a file : `gcc -o labserver labserver.c`
- Then use `./execName` to run the server.

1- Objective

1. Socket network programming
2. Review client/server approach.
   Each group (composed of 3 students at most) shall submit a report in campus.
   The report (PDF format is accepted) shall be uploaded on the campus page before the deadline November 22 at 23h55.
   The source code and the exec files of the server (websrv.c and websrv) as well as the index.html should be zipped in one file. Please propose a small html file.

2- Web server

1. Explain why we are using the library `<netinet/in.h >`

`<netinet/in.h >` is an internet protocol family, we are using it to include the `sockaddr` structure or to use `in_` that are defined with `typedef`.

2. Add the following line: `return EXIT_SUCCESS`, compile and execute the server code. What is the required Linux command line and the name of the Compiler.

The command line to compile is `gcc -o webserver webserver.c` and the command line to execute is `./webserver`. The compiler's name is GCC.

3. What would be the suitable domain in our case the domain, the type and the port number?

We are using `AF_INET` to use an IPV4 adress, `SOCK_STREAM` to use TCP and finally we can use the port 8080, port used for a service.

```
hints.ai_family = AF_INET;
hints.ai_socktype = SOCK_STREAM;
...
getaddrinfo(0, "8080", &hints, &bind_address);
```

4. Complete the instruction of line 37.

```
if (!ISVALIDSOCKET(socket_listen))
    {
        fprintf(stderr, "socket() failed. (%d)\n", GETSOCKETERRNO());
        return 1;
    }
```

5. Add the source code to manage the error of the binding function.

```
if (bind(socket_listen, bind_address->ai_addr, bind_address->ai_addrlen))
    {
        fprintf(stderr, "bind() failed. (%d)\n", GETSOCKETERRNO());
```

```
        return 1;
    }
```

6. Here you should add your code to manage the error of socket listening.

```
if (listen(socket_listen, 10) < 0)
    {
        fprintf(stderr, "listen() failed. (%d)\n", GETSOCKETERRNO());
        return 1;
    }
```

3- Connection test

7. Now, you will surround your accept function call by an infinite while loop (a server program should never quit until it is interrupted). Add the following code to make your server program able to serve an `index.html` file through the HTTP protocol. The `index.html` could be the simple html file.

```
#include <stdio.h>
#include <time.h>
#include <string.h>

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
#include <errno.h>
#include <fcntl.h>

#define ISVALIDSOCKET(s) ((s) >= 0)
#define CLOSESOCKET(s) close(s)
#define SOCKET int
#define GETSOCKETERRNO() (errno)

int main()
{
    struct addrinfo hints;
    memset(&hints, 0, sizeof(hints));
    hints.ai_family = AF_INET;
    hints.ai_socktype = SOCK_STREAM;
    hints.ai_flags = AI_PASSIVE;
    struct addrinfo *bind_address;
    getaddrinfo(0, "8080", &hints, &bind_address);
    printf("Creating socket...\n");
    SOCKET socket_listen;
    socket_listen = socket(bind_address->ai_family,
                           bind_address->ai_socktype,
                           bind_address->ai_protocol);
    if (!ISVALIDSOCKET(socket_listen))
    {
        fprintf(stderr, "socket() failed. (%d)\n", GETSOCKETERRNO());
        return 1;
    }
    //Binding
    printf("Binding socket to local address...\n");
    if (bind(socket_listen, bind_address->ai_addr, bind_address->ai_addrlen))
    {
        fprintf(stderr, "bind() failed. (%d)\n", GETSOCKETERRNO());
        return 1;
    }
    int file;
    char buf[BUFSIZ];
    int size;

    while (1)
    {
```

```
        //Listening
        listen(socket_listen, 0);
        //Acceptance
        SOCKET socket_client = accept(socket_listen, NULL, NULL);
        if (!ISVALIDSOCKET(socket_client))
        {
            fprintf(stderr, "accept() failed. (%d)\n", GETSOCKETERRNO());
            return 1;
        }
        size = read(socket_client, buf, BUFSIZ);
        write(1, buf, size);
        if ((file = open("index.html", O_RDONLY)) == -1)
        {
            printf("No existing file name `index.html`\n");
            return (1);
        }
        size = sprintf(buf, "HTTP/1.1 200 OK\n\n");
        size += read(file, buf + size, BUFSIZ);
        write(1, buf, size);
        write(socket_client, buf, size);
        close(socket_client);
        close(file);
    }
}
```