4,implement system call

Listen()=Marks a socket as passive to listen for incoming connecion

To implement the listen() system call, which marks a socket as passive and prepares it to accept incoming connections, this typically need to follow steps in a programming environment that supports socket programming, such as C with the POSIX socket API.

Implementing listen() =Marks a socket as passive to listen for incoming connection

Required Headers

Make sure to include the necessary headers for socket programming:

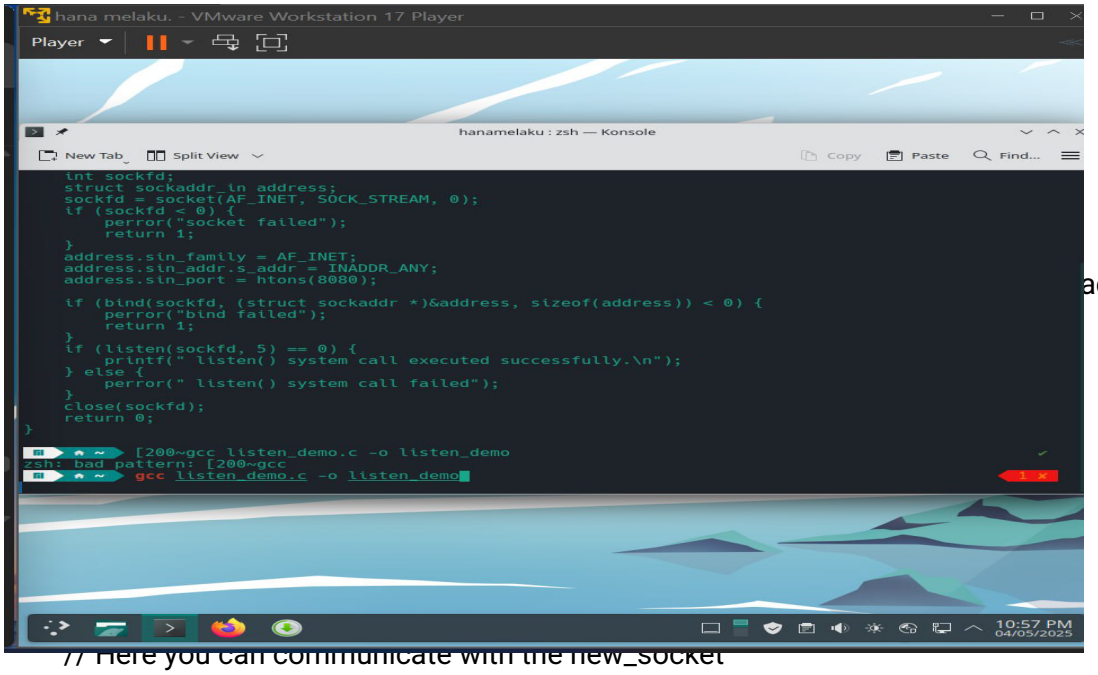
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
```

Main Server Code

```
#define PORT 8080 // Port number
#define BACKLOG 5 // Maximum number of pending connections

int main() {
    int server_fd, new_socket;
    struct sockaddr_in address;
    int opt = 1;
```

```
    int addrlen = sizeof(address);
    // Create socket file descriptor
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }
    // Set socket options
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt, sizeof(opt)))
{
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }
    // Define the address and port for the server
    address.sin_family = AF_INET; // IPv4
    address.sin_addr.s_addr = INADDR_ANY; // Accept connections from any IP
    address.sin_port = htons(PORT); // Convert port number to network byte order
    // Bind the socket to the specified address and port
    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }
    // Mark the socket as passive to listen for incoming connections
    if (listen(server_fd, BACKLOG) < 0) {
        perror("listen");
        exit(EXIT_FAILURE);
    }
```



addrlen)) < 0)

// Here you can communicate with the new_socket

// For simplicity, we will just close it after accepting

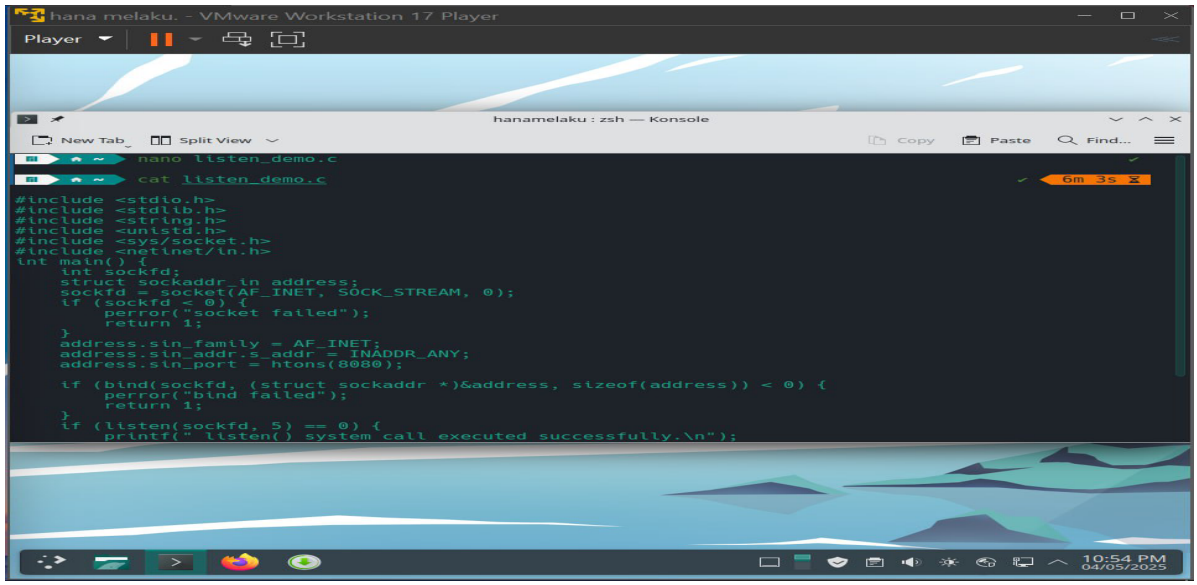
close(new_socket);

}

return 0;

}

Some images for the above system call is



Explanations of key steps about the above system call

1.create a socket :we create a socket using socket() with parameters specifiying IPv4 and TCP.

2. Set Socket Options: The setsockopt() function is used to set options for the socket. Here, we allow the socket to reuse the address and port.

3. Bind the Socket: The bind() function associates the socket with a specific IP address and port number.

4. Listen for Connections: The listen() function marks the socket as passive and specifies how many incoming connections can be queued (defined by BACKLOG).

5. Accept Connections: The server enters an infinite loop where it waits for incoming connections using accept(). When a connection is accepted, you can handle it (e.g., read/write data) before closing the connection.

To compile and run this code:

. Compile the Kernel:

- After making changes, you need to compile the kernel:

```
make -j$(nproc)
```

2. Install the Kernel:

- Install your new kernel:

```
sudo make modules_install
```

```
sudo make install
```

3. Reboot:

- Reboot your system into the new kernel.

4. Testing:

- Write a user-space application that creates a socket, binds it, and calls listen() to verify that your implementation works correctly.

Important Notes

- Modifying kernel code can lead to system instability or crashes. Make sure to test in a safe environment (like a virtual machine).

Make sure to run this on a system that supports POSIX sockets (like Linux or macOS). You can connect to this server using tools like telnet or create a corresponding client program.