Difference between C and C#

time.h localtime() function in C with Examples

asctime() and asctime_s() functions in C with Examples

return statement in C/C++ with Examples

size of char datatype and char array in C

Arrow operator -> in C/C++ with Examples

Logical Not ! operator in C with Examples

Socket Programming in C/C++: Handling multiple clients on server without multi threading

This tutorial assumes you have a basic knowledge of socket programming, i.e you are familiar with basic server and client model. In the basic model, server handles only one client at a time, which is a big assumption if you want to develop any scalable server model.

The simple way to handle multiple clients would be to spawn new thread for every new client connected to the server. This method is strongly not recommended because of various disadvantages, namely:

- Threads are difficult to code, debug and sometimes they have unpredictable results.
- Overhead switching of context

- Not scalable for large number of clients
- Deadlocks can occur

Select()

A better way to handle multiple clients is by using **select()** linux command.

- Select command allows to monitor multiple file descriptors, waiting until one of the file descriptors become active.
- For example, if there is some data to be read on one of the sockets select will provide that information.
- **Select** works like an interrupt handler, which gets activated as soon as any file descriptor sends any data.

Data structure used for select: fd_set

It contains the list of file descriptors to monitor for some activity.
There are four functions associated with fd_set:

```
fd_set readfds;

// Clear an fd_set
FD_ZERO(&readfds);

// Add a descriptor to an fd_set
FD_SET(master_sock, &readfds);

// Remove a descriptor from an fd_set
FD_CLR(master_sock, &readfds);

//If something happened on the master socket , then its an incoming conne
FD_ISSET(master_sock, &readfds);
```

Activating select: Please read the man page for select to check all the arguments for select command.

```
activity = select( max_fd + 1 , &readfds , NULL , NULL , NULL);
```

Implementation:

```
//Example code: A simple server side code, which echos back the receiv
//Handle multiple socket connections with select and fd_set on Linux
#include <stdio.h>
#include <string.h>   //strlen
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>   //close
#include <arpa/inet.h>    //close
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <sys/time.h> //FD_SET, FD_ISSET, FD_ZERO macros

#define TRUE   1
#define FALSE  0
#define PORT 8888

int main(int argc , char *argv[])
{
    int opt = TRUE;
    int master_socket , addrlen , new_socket , client_socket[30] ,
          max_clients = 30 , activity, i , valread , sd;
    int max_sd;
    struct sockaddr_in address;

    char buffer[1025];  //data buffer of 1K

    //set of socket descriptors
    fd_set readfds;

    //a message
    char *message = "ECHO Daemon v1.0 \r\n";

    //initialise all client_socket[] to 0 so not checked
    for (i = 0; i < max_clients; i++)
    {
        client_socket[i] = 0;
    }

    //create a master socket
    if( (master_socket = socket(AF_INET , SOCK_STREAM , 0)) == 0)
    {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    //set master socket to allow multiple connections ,
    //this is just a good habit, it will work without this
    if( setsockopt(master_socket, SOL_SOCKET, SO_REUSEADDR, (char *)&c
          sizeof(opt)) < 0 )
    {
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }

    //type of socket created
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
```

Compile the file and run the server.
Use telnet to connect the server as a client.

Try running on different machines using following command:

```
telnet localhost 8888
```

Code Explanation:

- We have created a fd_set variable readfds, which will monitor all the active file descriptors of the clients plus that of the main server listening socket.
- Whenever a new client will connect, master_socket will be activated and a new fd will be open for that client. We will store its fd in our client_list and in the next iteration we will add it to the readfds to monitor for activity from this client.
- Similarly, if an old client sends some data, readfds will be activated and we will check from the list of existing client to see which client has send the data.

Alternatives:
There are other functions that can perform tasks similar to select. pselect , poll , ppoll

This article is contributed by **Akshat Sinha**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recommended Posts:

Determinant of N x N matrix using multi-threading
Linear search using Multi-threading
Maximum in a 2D matrix using Multi-threading in C++
Socket Programming in C/C++
Sort an array using socket programming in C
What is web socket and how it is different from the HTTP?
Explicitly assigning port number to client in Socket
How to write long strings in Multi-lines C/C++?
Multi-set for user defined data type
Difference between Web Browser and Web Server
TCP Server-Client implementation in C
UDP Server-Client implementation in C

JSF | Java Server Faces

How to install Apache server in Ubuntu ?

How slow HTTP can knock down a server?

Article Tags : C C++ GBlog

Practice Tags : C CPP

Be the First to upvote.

3.2

To-do Done

Based on **5** vote(s)

Feedback/ Suggest Improvement Add Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments