

## DATE AND TIME SERVER USING SOCKET

Aim: To create a server that continuously runs and sends the date and time as soon as a client connects to it.

### ALGORITHM:

- 1) Create a new socket of family/domain AF\_INET and the transport layer protocol as TCP.
- 2) Bind the socket with a structure that contains details such as address, port number.

2) Find the socket with a structure that contains details such as interface, port etc.

3) Call 'listen()' to start listening process of socket.

4) As soon as some receives request from a client, it prepares the date and time and writes on the client socket through the description returned by accept().

Source code:

```
#include <sys/types.h>
```

```
#include <netinet/in.h>
```

```
#include <arpa/inet.h>
```

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

```
#include <stdlib.h>
```

```
#include <errno.h>
```

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

```
#include <sys/types.h>
```

```
#include <unistd.h>
```

```
int main(void) {
```

```

int main ( int argc, char * argv[] )
{
    int    fd; int opt = 0; int port = 0;
    struct sockaddr_in serv_addr;
    char    sendBuff[1024];
    int     len;
    listen fd = socket ( AF_INET, SOCK_
    STREAM, 0 );
    memset ( serv_addr, '0', sizeof ( serv_addr ) );
    memset ( sendBuff, '0', sizeof ( sendBuff ) );
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl (
    INADDR_ANY );

```



```

ServerAddr = port = htons(5000);
len = (sizeof(struct sockaddr_in) + sizeof(char *));
sock = socket(AF_INET, SOCK_STREAM, 0);
bind(sock, (struct sockaddr *)&ServerAddr, len);
listen(sock, 5);
while(1) {
    struct sockaddr_in clientAddr;
    int clientLen = sizeof(clientAddr);
    socklen_t len = sizeof(clientAddr);
    fd = accept(sock, (struct sockaddr *)&clientAddr, &len);
    if(fd == -1) continue;
    printf("Got Buff, sizeof (Send Buff) = %d\n", sizeof(SendBuff));
    write(fd, SendBuff, sizeof(SendBuff));
    close(fd);
    sleep(2);
}

```

Print code

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <netdb.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <arpa/inet.h>

int main(int argc, char \*argv[])

```

int sockfd = 0, n = 0;
char recvBuff[1024];
struct sockaddr_in serv_addr;
if (argc != 2) {
    printf("Usage: %s <ip of server>\n",
        argv[0]);
    return 1;
}
memset(recvBuff, '0', sizeof(recvBuff));
if ((sockfd = socket(AF_INET, SOCK_STREAM, 0))
    < 0) {
    printf("Error: could not create socket\n");
}

```

```

return 1, 3;
memset(&serv_addr, '0', sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(8080);
if (inet_pton(AF_INET, argv[1], &serv_
addr.sin_addr) < 0) {
printf("inet_pton error occured\n");
return 1, 3;
}
while (1) {
if (read(sockfd, recvBuff, sizeof
(recvBuff)) > 0)

```



```

while ((w = read (buffd, recvBuff, sizeof
(recvBuff) - 1)) > 0)
{
    recvBuff[n] = 0;
    if (fgets(recvBuff, sizeof(recvBuff), stdin) == EOF) {
        printf("\n Error: Fgets error\n");
    }
    if (n < 0) {
        printf("\n Read error\n");
    }
    return 0;
}

```

0/8:

4. / nurse 127.0.0.1

4.11 Apr 23 22:22:14 2016