ANNA UNIVERSITY, GUINDY, CHENNAI:: 600 025
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course Code: CS6111

Course Name: Computer Networks

Semester : V

Batch : N&Q

LAB EXAM **DATE:30.08.2024**

1.Write client-server application in C using Sockets

The Server Create a logging server that receives (from a client) requests log messages to the log file or to retrieve the content of the file.

The request consists of the following form:

1. ADD# (Message]

Message: is the message to be logged in the log file. This command will write the Message to the log file and should return the following message to the client

Message Logged Successfully

2. GET! (N)

N: the number of messages to be retrieved from the log file

This command returns the first N messages to the client. Each message contains only one line and messages should be separated by #.

Write a main program that creates an object of the Server Socket and indefinitely listen for the coming connections, once the connection accepted the client should be able to communicate the commands ADD and GET .

2.
Use the given program for Connection-Oriented Iterative service in which server calculates the Net-salary of an Employee based on the following basic salary details sent by the client

calculate i) hra -10% of Basic Sal ii) da- 25% iii.epf -2% iv. (net-sala=basic+hra\-epf)

Find the missing parameters and values in the given program and complete the program.

Simple Stream Oriented Server

A Simple Stream-Oriented Server
Uses TCP Port Number 3456

```
#include <stdio>;
#include <stdlib>;
#include <errno>;
#include <string>;
#include <sys/types>;
#include <netinet/in>;
#include <sys/socket>;
#include <sys/wait>;
#define MYPORT    /* the port users will be connecting to */
#define BACKLOG 10 /* number of pending connections */
main()
{
int sockfd, new_fd; /* listen on sock_fd,new connection on new_fd */
struct sockaddr_in my_addr; /* my address information */
struct sockaddr_in their_addr; /* client's address info */
int sin_size;
if ((sockfd = socket() == -1)
{
perror("socket");
exit(1);
}
my_addr.sin_family = ;
my_addr.sin_port = ;
my_addr.sin_addr.s_addr = ; /* auto-fill with my IP */
bzero(&(my_addr.sin_zero), 8); /* zero the rest */
if (bind(sockfd,(struct sockaddr *)&my_addr, )
{
perror("bind");
exit(1);
}
if (listen(sockfd, BACKLOG) == -1)
{
perror("listen");
exit(1);
}
while(1) /* main accept() loop */
{
sin_size = sizeof(struct sockaddr_in);
```

```
if ((new_fd = accept(sockfd,(struct sockaddr *)&their_addr,&sin_size)) ==
-1)
{
perror("accept");
continue;
}
printf("server: got connection from %s\n",
inet_ntoa(their_addr.sin_addr));
if (!fork()) /* this is the child process */
{
if (send( , "Hello, world!\n", ,) == -1)
perror("send");
close(new_fd);
exit(0);
}
close(new_fd); /* parent doesn't need this */
while(waitpid(-1,NULL,WNOHANG) > 0); /* clean up child processes */
}
}
```

Simple Stream Oriented Client

```
#include <stdio>;
#include <stdlib>;
#include <errno>;
#include <string>;
#include <netdb>;
#include <sys/types>;
#include <netinet/in>;
#include <sys/socket>;
#define PORT   /* the port client will be connecting to */
#define MAXDATASIZE 100 /* max number of bytes we can get at once */
int main(int argc, char *argv[])
{
int sockfd, numbytes;
char buf[MAXDATASIZE];
struct hostent *he;
struct sockaddr_in their_addr; /* client's address information */
if (argc != 2)
{
fprintf(stderr,"usage: client hostname\n");
exit(1);
}
if ((he=gethostbyname(argv[1])) == NULL) /* get the host info */
{
herror("gethostbyname");
```

```
exit(1);
}
if ((sockfd =  == -1)
{
perror("socket");
exit(1);
}
their_addr.sin_family = ;
their_addr.sin_port = ;
their_addr.sin_addr = ;
bzero(&(their_addr.sin_zero), 8);
if (connect(sockfd,()) ==-1)
{
perror("connect");
exit(1);
}
if ((numbytes=recv()) == -1)
{
perror("recv");
exit(1);
}
buf[numbytes] = '\0';
printf("Received: %s",buf);
close(sockfd);
return 0;
}
```