assistant.c

```
#include
#include
#include
#include
#include
#include
#include
#include
#include
#include
#include "defs.h"
#include
#include
#include
#include
#define FILEPATH "History.txt"

//Group A
//Nikul Patel - worked on what to do after query was sent to server, and
termPrinter function
//nikul.patel@okstate.edu
//CWID: A20066980

//void termPrinter(char *results);
bool equalsIgnoreCase(char* str1, char* str2);
FILE *fterm;
int LinePosition = 1;
char path[20] = "/dev/pts/";
bool isFirst = TRUE;


void assistant(){
//Pipe variables
int fd;
char pipe[] = ".pipe";

//History file variables
FILE* ftemp;
FILE* infile = fopen("History.txt","r");
char nameQuery[256] = {0}, jobQuery[256] = {0}, statusQuery[256] = {0};
char job[256] = {0}, name[256] = {0}, status[256] = {0};
char buffer[1024] = {0}, query[1024] = {0};
bool foundMatch = FALSE;
float id = 0;
int count = 0;
bool history = FALSE;
char incomingBuffer[256];
```

```
/* create the FIFO (named pipe) */
mkfifo(pipe, 0666);

// Open pipe as read only for the assistant
if((fd = open(pipe, O_RDONLY | O_CREAT)) < 0) perror("Pipe creation failure");

// Socket creation
int network_socket = socket(AF_INET,SOCK_STREAM, 0);
struct sockaddr_in server_address;
server_address.sin_family = AF_INET;
server_address.sin_port = htons(9002);
server_address.sin_addr.s_addr = INADDR_ANY;

//Connect to the server and verify that the connection succeeded
int connection_status = connect(network_socket, (struct sockaddr*)&server_address,
sizeof(server_address));
if (connection_status == -1){
perror("There was an error establishing a connection");
exit(EXIT_FAILURE);
}

// check number of lines written in History.txt
int LineCount = 0;
char storeC;
char lineBuffer[256];
if(infile == NULL){
printf("unable to open file\n");
exit(EXIT_FAILURE);
}

while((storeC = fgetc(infile)) != EOF){
if(storeC == '\n'){
LineCount++;
}
}


while(TRUE){
//Get query from manager through the pipe
if(read(fd, &query, sizeof(query) + 1) < 0) perror("Read failure");
sscanf(query, "%[^,]%*c%[^,]%*c%[^\n]%*c", nameQuery, jobQuery, statusQuery);
//Check for info in History.txt first

if(!historySearch(query)){
//Send name to server to see if it contains the employee
send(network_socket, nameQuery, 256, 0);
recv(network_socket, incomingBuffer, sizeof(incomingBuffer), 0);
//Wait for a non empty message to be received.
while (strcmp(incomingBuffer, "") == 0) {
recv(network_socket, incomingBuffer, sizeof(incomingBuffer), 0);
}
if(strcmp(incomingBuffer, "INVALID")==0){}
//if file has 10 lines use position to overwrite records
else if(LineCount == 10){
//program uses temp.txt to store records from History.txt
infile = fopen("History.txt", "r");
ftemp = fopen("temp.txt", "w");
while((fgets(lineBuffer, 256, infile)) != NULL){
count++;
if(count == LinePosition){
fprintf(ftemp, "%s\n", incomingBuffer);
}else{
fprintf(ftemp, "%s", lineBuffer);
}
}

fclose(infile);
fclose(ftemp);
//replace History.txt with the temp.txt
remove("History.txt");
rename("temp.txt", "History.txt");
LinePosition++;
if(LinePosition == 11){
LinePosition = 1;
}
}else{
infile = fopen("History.txt", "a");
fprintf(infile, "%s\n", incomingBuffer);
fclose(infile);
}
termPrinter(incomingBuffer);
}
}
close(network_socket);
}


//function that prints to a new terminal
//requires tty ID for printing
void termPrinter(char *results){
if(isFirst){
printf("please type terminal tty ID number(type tty into the terminal you wish to
print on and enter the last digit shown)\n");
char id[2];
scanf("%[^\n]%*c", id);
strcat(path, id);
printf("%s\n", path);
isFirst = FALSE;
}

fterm = fopen(path, "wr");
```

```
if(fterm != NULL){
fprintf(fterm, "%s\n", results);
}else{
perror("tty terminal not found");
exit(EXIT_FAILURE);
}

fclose(fterm);
}
```