The server program below waits for new connections from multiple clients. If a connection occurs, server program uses fork() system call to create a child process for handling the new connection.

The client program below sends “ls” string to the server. When the server receives a message from connected clients, it checks if the received message is “ls”. If it is, it executes ls command and sends the output back to the client.

Modify the codes below to accomplish the following tasks:

- Currently, client program terminates after one send - recv operation. Let the program run until “end” message.
- Add additional commands (e.g. cat, cp...)

```c
/*
 * ============================================================================
 * Server
 * ============================================================================
 */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/wait.h>
#define MYPORT 3490 // the port users will be connecting to
#define BACKLOG 10 // how many pending connections queue will hold
#define MAXDATA 1000
void fileServer(int sockID) {
	FILE *f;
	int n;
	char buf[MAXDATA];
	n = recv(sockID, buf, 10, 0);
	buf[n] = '\0';
	if (strcmp(buf, "ls") == 0) {
		system("ls > templs");
		f = fopen("templs", "r");
		n = fread(buf, 1, MAXDATA, f);
		send(sockID, buf, n, 0);
		fclose(f);
	}
	close(sockID);
}
int main() {
	int sockfd, new_fd, pid;
	struct sockaddr_in my_addr; // my address information
	struct sockaddr_in their_addr; // connector’s address information
	int sin_size;
	int yes = 1;
	if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
		perror("socket");
		exit(1);
	}
	setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(int));
	my_addr.sin_family = AF_INET; //Internet Address Family (IP)
	my_addr.sin_port = htons(MYPORT); // short, network byte order
	my_addr.sin_addr.s_addr = INADDR_ANY;
	memset(&(my_addr.sin_zero), '\0', 8);
	if (bind(sockfd, (struct sockaddr *) &my_addr, sizeof(struct sockaddr)) == -1) {
		perror("bind");
		exit(1);
	}
	if (listen(sockfd, BACKLOG) == -1) {
		perror("listen");
		exit(1);
	}
	sin_size = sizeof(struct sockaddr_in);
	while (1) {
		if ((new_fd = accept(sockfd, (struct sockaddr *) &their_addr, &sin_size)) == -1) {
			perror("accept");
			exit(1);
		} else {
			pid = fork();
			if (pid == 0)  // if in child process
				fileServer(new_fd);
		}
	}
	return 0;
}
```

```c
/*
 * ================================================================
 * Client
 * ================================================================
 */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#define PORT 3490
#define MAXDATASIZE 1000 // max number of bytes we can get at once
int main() {
        char IP[16];
        int sockfd, numbytes;
        char buf[MAXDATASIZE];
        struct sockaddr_in server; // connector’s address information
        if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
                perror("socket");
                exit(1);
        }
        server.sin_family = AF_INET; // Internet Address Family (IP)
        server.sin_port = htons(PORT); // short, network byte order
        printf("\n\n\nEnter IP address of the Server\n");
        scanf("%s", IP);
        server.sin_addr.s_addr = inet_addr(IP);
        memset(&(server.sin_zero), '\0', 8);
        if (connect(sockfd, (struct sockaddr *) &server, sizeof(struct sockaddr)) == -1) {
                perror("connect");
                exit(1);
        }
        send(sockfd, "ls", 2, 0);
        if ((numbytes = recv(sockfd, buf, MAXDATASIZE - 1, 0)) == -1) {
                perror("recv");
                exit(1);
        }
        buf[numbytes] = '\0';
        printf("Received: %s", buf);
        close(sockfd);
        return 0;
}
```