PROGRAM CODE

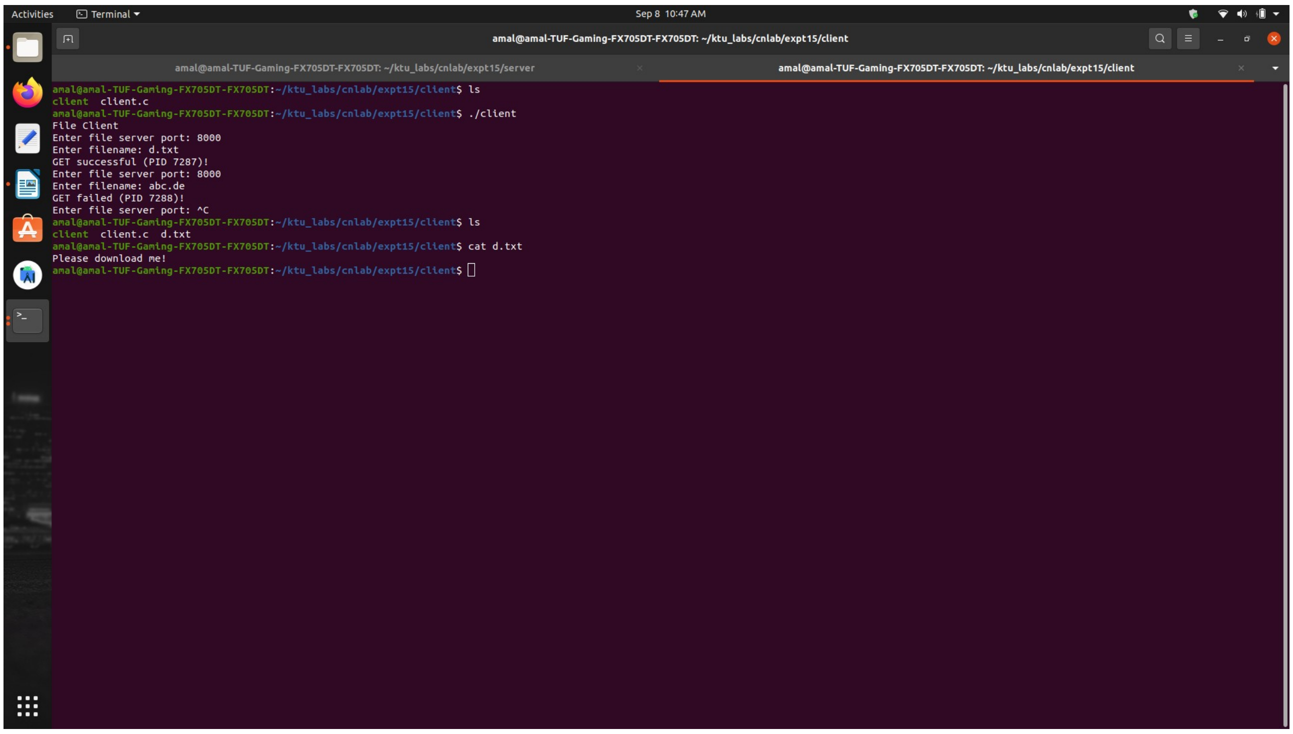
server.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
#include <dirent.h>

void serverRecv(int client_fd) {
	char str[100];
	char str1[1000];

	if(recv(client_fd, str, 100 * sizeof(char), 0) <= 0) {
		printf("Connection lost!\n");
		return;
	}

	FILE* fp = fopen(str, "r");
	int pid = getpid();

	if(fp == NULL) {
		str1[0] = '\0';

		if(send(client_fd, str1, sizeof(char), 0) <= 0) {
			printf("GET failed!\n");
			return;
		}

		if(send(client_fd, &pid, sizeof(int), 0) <= 0) {
			printf("GET failed!\n");
			return;
		}

		if(send(client_fd, "400 BAD", 8 * sizeof(char), 0) <= 0) {
			printf("GET failed!\n");
			return;
		}

		return;
	}

	str1[0] = '\0';

	while(fgets(str, 100, fp) != NULL) {
		strcat(str1, str);
	}

	fclose(fp);

	if(send(client_fd, str1, 1000 * sizeof(char), 0) <= 0) {
		printf("GET failed!\n");
		return;
	}

	if(send(client_fd, &pid, sizeof(int), 0) <= 0) {
		printf("GET failed!\n");
		return;
	}
	if(send(client_fd, "200 OK", 7 * sizeof(char), 0) <= 0) {
		printf("GET failed!\n");
		return;
	} else {
		printf("GET successful!\n");
		return;
	}
}

void main(int argc, char* argv[]) {
	int PORT;

	if(argc == 2) {
		PORT = atoi(argv[1]);
	} else {
		printf("Enter file server port!\n");
		exit(1);
	}

	int server_fd, client_fd;
	struct sockaddr_in address;
	int addrlen = sizeof(address);

	printf("Concurrent File Server\n");

	if((server_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
		printf("Socket creation failed!\n");
		exit(1);
	}

	address.sin_family = AF_INET;
	address.sin_addr.s_addr = INADDR_ANY;
	address.sin_port = htons(PORT);

	if(bind(server_fd, (struct sockaddr*) &address, addrlen) < 0) {
		printf("Socket binding failed!\n");
		exit(1);
	}

	if(listen(server_fd, 5) < 0) {
		printf("Listening failed!\n");
		exit(1);
	}

	while(1) {
		if((client_fd = accept(server_fd, (struct sockaddr*) &address, 
(socklen_t*) &addrlen)) < 0) {
			printf("Connection failed!\n");
			exit(1);
		} else {
			printf("Connected to client.\n");
		}

		if(fork() == 0) {
			serverRecv(client_fd);

			close(client_fd);
		}
	}

	close(server_fd);
}
```

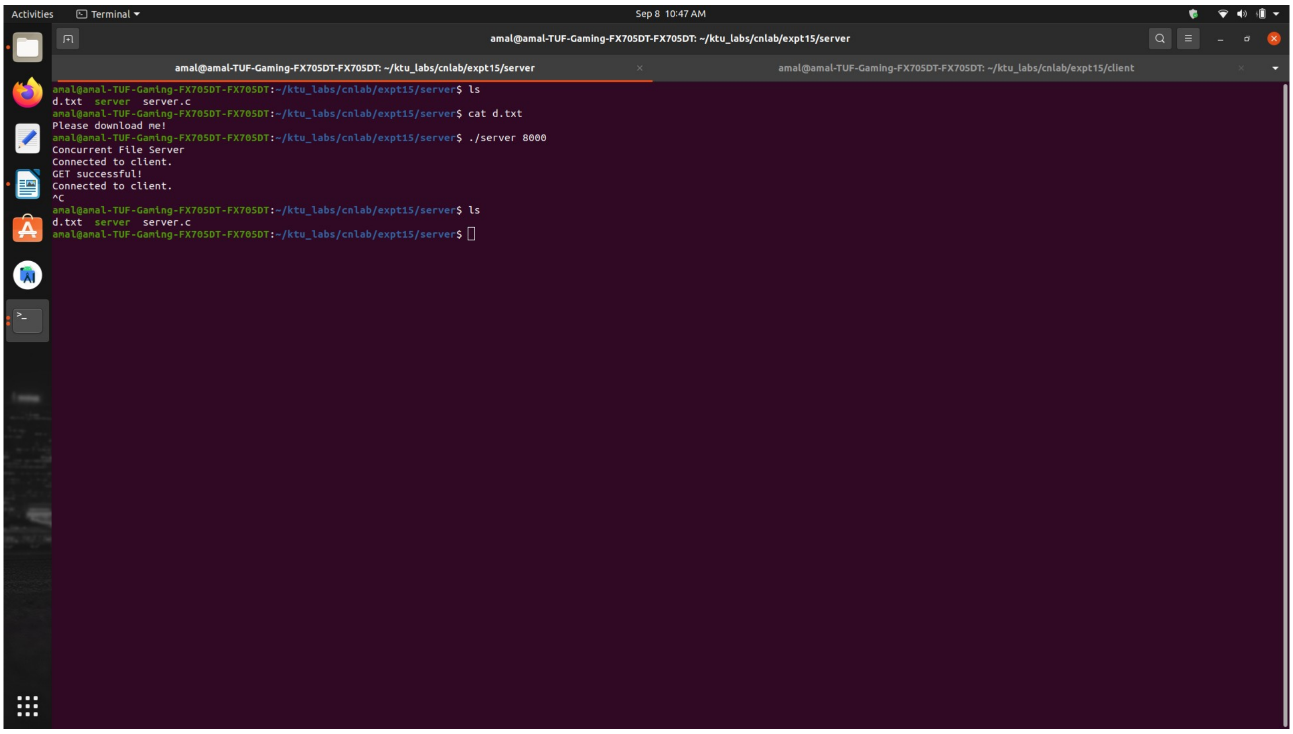
client.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>

void get(int client_fd) {
      char filename[100];
      char str[100];
      char str1[1000];
      printf("Enter filename: ");
      scanf("%s", filename);

      if(send(client_fd, filename, 100 * sizeof(char), 0) <= 0) {
            printf("GET failed!\n");
            return;
      } else {
            if(recv(client_fd, str1, 1000 * sizeof(char), 0) <= 0) {
                  printf("GET failed!\n");
                  return;
            }

            int pid;

            if(recv(client_fd, &pid, sizeof(int), 0) <= 0) {
                  printf("GET failed!\n");
                  return;
            }

            if(recv(client_fd, str, 100 * sizeof(char), 0) <= 0) {
                  printf("GET failed!\n");
                  return;
            }

            if(!strcmp(str, "200 OK")) {
                  FILE* fp = fopen(filename, "w");
                  fputs(str1, fp);

                  printf("GET successful (PID %d)!\n", pid);

                  fclose(fp);
            } else {
                  printf("GET failed (PID %d)!\n", pid);
            }
      }
}

void main() {
      int client_fd, PORT;
      struct sockaddr_in serv_addr;

      printf("File Client\n");

      serv_addr.sin_family = AF_INET;
      serv_addr.sin_addr.s_addr = INADDR_ANY;

      while(1) {
            printf("Enter file server port: ");
            scanf("%d", &PORT);
            serv_addr.sin_port = htons(PORT);

            client_fd = socket(AF_INET, SOCK_STREAM, 0);

            if(client_fd < 0) {
                  printf("Socket creation failed!\n");
                  exit(1);
            }

            if(connect(client_fd, (struct sockaddr*) &serv_addr,
sizeof(serv_addr)) < 0) {
                  printf("Connection failed!\n");
                  exit(1);
            }

            get(client_fd);

            close(client_fd);
      }
}
```

OUTPUT


Activities
Terminal
Sep 8 10:47 AM
amal@amal-TUF-Gaming-FX705DT-FX705DT: ~/ktu_labs/cnlab/expt15/client
amal@amal-TUF-Gaming-FX705DT-FX705DT: ~/ktu_labs/cnlab/expt15/server
amal@amal-TUF-Gaming-FX705DT-FX705DT: ~/ktu_labs/cnlab/expt15/client

```
amal@amal-TUF-Gaming-FX705DT-FX705DT:~/ktu_labs/cnlab/expt15/client$ ls
client  client.c
amal@amal-TUF-Gaming-FX705DT-FX705DT:~/ktu_labs/cnlab/expt15/client$ ./client
File Client
Enter file server port: 8000
Enter filename: d.txt
GET successful (PID 7287)!
Enter file server port: 8000
Enter filename: abc.de
GET failed (PID 7288)!
Enter file server port: ^C
amal@amal-TUF-Gaming-FX705DT-FX705DT:~/ktu_labs/cnlab/expt15/client$ ls
client  client.c  d.txt
amal@amal-TUF-Gaming-FX705DT-FX705DT:~/ktu_labs/cnlab/expt15/client$ cat d.txt
Please download me!
amal@amal-TUF-Gaming-FX705DT-FX705DT:~/ktu_labs/cnlab/expt15/client$ 
```


Activities
Terminal
Sep 8 10:47 AM
amal@amal-TUF-Gaming-FX705DT-FX705DT: ~/ktu_labs/cnlab/expt15/server
amal@amal-TUF-Gaming-FX705DT-FX705DT: ~/ktu_labs/cnlab/expt15/server
amal@amal-TUF-Gaming-FX705DT-FX705DT: ~/ktu_labs/cnlab/expt15/client

```
amal@amal-TUF-Gaming-FX705DT-FX705DT:~/ktu_labs/cnlab/expt15/server$ ls
d.txt  server  server.c
amal@amal-TUF-Gaming-FX705DT-FX705DT:~/ktu_labs/cnlab/expt15/server$ cat d.txt
Please download me!
amal@amal-TUF-Gaming-FX705DT-FX705DT:~/ktu_labs/cnlab/expt15/server$ ./server 8000
Concurrent File Server
Connected to client.
GET successful!
Connected to client.
^C
amal@amal-TUF-Gaming-FX705DT-FX705DT:~/ktu_labs/cnlab/expt15/server$ ls
d.txt  server  server.c
amal@amal-TUF-Gaming-FX705DT-FX705DT:~/ktu_labs/cnlab/expt15/server$ 
```