

Introdução

Bom, inicialmente eu segui as vídeo aulas presentes no moodle para criar uma comunicação padrão. Como havia especificações (como não enviar o caracter \o, o servidor não precisar se conectar a mais de um cliente por vez, etc) eu acabei fazendo algumas pequenas modificações na comunicação padrão.

Após isso eu fiz a criação de uma função, func (), no serve, para deixar o código mais organizado.

```
C server.c u x C client.c u Makefile u
C server.c > func(int)
56     if (0 != listen(s, 10)) {
57         |     logexit("listen");
58     }
59
60     char addrstr[BUFSZ];
61     addrtostr(addr, addrstr, BUFSZ);
62     printf("bound to %s, waiting connections\n", addrstr);
63     func(s);
64     close(s);
65     exit(EXIT_SUCCESS);
66 }
67
68 void func(int s) {
69     char messageReceive[BUFSZ];
70     char messageSend[BUFSZ];
```

Nessa função eu verificava o comando recebido pelo servidor, inclusive comandos inválidos, e direcionada a respectiva função, de acordo com o proposto pelo TP.

```
C server.c u x C client.c u Makefile u
C server.c > func(int)
68 void func(int s) {
69     char messageReceive[BUFSZ];
70     char messageSend[BUFSZ];
71     char *command;
72     clear_bd();
73
74     while (1) {
75         struct sockaddr_storage cstorage;
76         struct sockaddr *caddr = (struct sockaddr *)&cstorage;
77         socklen_t caddrlen = sizeof(cstorage);
78
79         int csock = accept(s, caddr, &caddrlen);
80         if (csock == -1) {
81             |     logexit("accept");
82         }
83
84         char caddrstr[BUFSZ];
85         addrtostr(caddr, caddrstr, BUFSZ);
86         printf("[log] connection from %s\n", caddrstr);
87
88         size_t count = recv(csock, messageReceive, BUFSZ - 1, 0);
89         printf("[msg] %s, %d bytes: %s\n", caddrstr, (int)count,
90             |     messageReceive);
91
92         char buf[BUFSZ];
93         memset(buf, 0, BUFSZ);
94
95         if (strncmp(messageReceive, "add", 3) == 0) {
96             bzero(messageSend, BUFSZ);
97             strcpy(messageSend, add(messageReceive));
98
99             count = send(csock, messageSend, strlen(messageSend), 0);
100             if (count != strlen(messageSend)) {
101                 |     logexit("send");
102             }
103             close(csock);
104         }
105
106         else if (strncmp(messageReceive, "remove", 6) == 0) {
107             bzero(messageSend, BUFSZ);
108             strcat(messageSend, remov(messageReceive));
109             count = send(csock, messageSend, strlen(messageSend), 0);
110             if (count != strlen(messageSend)) {
111                 |     logexit("send");
112             }
113         }
```

```

106     }
107
108     else if (strncmp(messageReceive, "remove", 6) == 0) {
109         bzero(messageSend, BUFSZ);
110         strcat(messageSend, remov(messageReceive));
111         count = send(csock, messageSend, strlen(messageSend), 0);
112         if (count != strlen(messageSend)) {
113             logexit("send");
114         }
115         close(csock);
116
117     } else if (strncmp(messageReceive, "list", 4) == 0) {
118         bzero(messageSend, BUFSZ);
119         strcat(messageSend, list(messageReceive));
120         count = send(csock, messageSend, strlen(messageSend), 0);
121         if (count != strlen(messageSend)) {
122             logexit("send");
123         }
124         close(csock);
125
126     } else if (strncmp(messageReceive, "read", 4) == 0) {
127         bzero(messageSend, BUFSZ);
128         strcat(messageSend, reader(messageReceive));
129         count = send(csock, messageSend, strlen(messageSend), 0);
130         if (count != strlen(messageSend)) {
131             logexit("send");
132         }
133         close(csock);
134
135     } else if (strncmp(messageReceive, "kill", 4) ==
136                 0) { // if msg contains "kill" then server exit and chat
137                     // ended.
138         close(csock);
139         break;
140     } else {
141         break;
142     }
143 }
144

```

Eu usei uma matriz 5x5 para evitar o índice 0. Essa matriz era global para poder ser acessada por todas as funções. Cada função recebia e retornava uma string de 500bits.

Um dos desafios enfrentados por mim inicialmente foi que eu não sabia que eu tinha que especificar um tamanho para a string de retorno das minhas funções, e isso me custou muito tempo até eu descobrir que tinha que realizar a alocação dinâmica, até então eu me deparava com segmentation fault.

Eu também tive muita dificuldade em construir a função reader por causa dos tratamentos envolvendo sensores existentes nos equipamentos e sensores não existentes.

Nas minhas funções eu usei a função strtok para quebrar meus parâmetros recebidos. Eu colocava esses tokens em outras variáveis para conseguir usar partes específicas da mensagem recebida.