Diogo Oliveira Neiss

Execução

Juntamente do makefile, criei dois arquivos .sh para compilar e executar o servidor, de modo a emular um “hot reload”, e outro para apenas executar o cliente, já que ele não muda muito. Ambos já passam uma porta específica e versão de protocolo.

Estratégia de implementação

Para implementar os dois sistemas, utilizei um mesmo socket para enviar mensagens no cliente e um mesmo socket para receber no servidor, sem ser necessário re-chamar a conexão.

No que tange a implementação do servidor, utilizei armazenamento com uma lista encadeada simples. Um problema recorrente era a remoção de itens no meio da lista e, para resolver isso, criei uma flag “deleted” dentro de cada instância de local de vacinação, de modo a saber se ela foi apagada ou não. Futuramente isso pode ser aperfeiçoado utilizando um método de limpeza, caso o volume de dados deletados seja grande, porém essa abordagem tem a vantagem de permitir que seja sabido quais dados forem deletados, ou seja, a remoção é apenas em termos de funcionalidade, não permanente.

Uma vez recebida a entrada, como “add 1 1”, utilizo uma função para separar a string em n partes, como “add”, “1” e “1”. Com as partes em mãos, consigo definir qual comando chamar e seus parâmetros. Deveria ter implementado essa mesma separação para múltiplas instruções numa mesma chamada, porém não deu tempo.

Os métodos de remoção e adição trabalham buscando primeiro se o local está presente na lista, e age de acordo se for o caso. Depois, no caso oposto, outra ação é tomada, e o que foi feito é retornado através de um inteiro, devidamente documentado na definição da função.

O método de listagem percorre a lista, adicionando a um buffer todos os locais que não tiverem sido marcados com a flag de deletado, depois retornando para ser enviado de resposta

O método de query trabalha calculando a menor distância atual, até o final da lista, onde a menordistância atual se tornará a absoluta. Não foi necessário calcular raízes quadradas, uma vez que uso a distância apenas comparativamente.

O método kill é o primeiro a ser lido, fechando a conexão imediatamente e saído do loop de recebimento de mensagens.

Desafios

O primeiro grande desafio foi a modelagem de dados. Gastei um tempo considerável utilizando array simples, para depois perceber e ter que recorrer a lista ligadas. Depois, outro ponto complicado foram os comportamentos inesperados do servidor no envio de mensagens, muitas vezes não enviando a mensagem mesmo que ela seja calculada corretamente, algo que pretendo resolver na próxima entrega da tarefa.

Imprevistos

Um imprevisto não calculado foi o comportamento das strings em c, principalmente quando há problemas de tamanho/ terminação. Para resolver isso, invés de utilizar funções padrão, como strtok, preferi utilizar uma biblioteca externa, já testada e com uma camada de abstração maior, evitando erros, assim como uma implementação de lista ligada.

Outro imprevisto foi a implementação do professor Ítalo, que servia para requisitos diferentes deste trabalho, o que levou a necessidade de modificar seu comportamento extensivamente, de modo que o que mais se aproveitou foram as definições e conexões de rede.

Por fim, e provavelmente o maior imprevisto de todos: a função recv não funcionava corretamente se eu aguardava o retorno dela ser 0, esperando infinitamente. Ainda não achei uma solução para isso, então ficará pendente para a próxima solução.

Dependências externas utilizadas

Biblioteca de separação de strings, disponível em https://github.com/mr21/strsplit.c

Lista ligada, disponível em https://gist.github.com/ArnonEilat/4470948

Código base do projeto, disponível no moodle/playlist do professor Ítalo.

Próximos passos

Planejo debugar extensivamente a aplicação, utilizando softwares mais robustos, como o Clion, de modo a entender linha a linha o que está acontecendo. Utilizei o gdb para investigar falhas de memória, porém não sou proficiente o suficiente nele para usar no projeto todo.

Planejo também fazer com que todos os métodos funcionem corretamente em seus retornos, não apenas conceitualmente, garantindo a aprovação nos testes.