Trabalho Prático 1 - Redes de Computadores

Matheus Filipe Sieiro Vargas - 2014144812

Setembro 2020

Protocolo de comunicação

Para estabelecer a comunicação entre cliente e servidor, foi utilizado o código *single thread* disponibilizado como material de apoio, devidamente alterado para possibilitar o envio das mensagens de acordo com o protocolo estabelecido na especificação do trabalho.

As mensagens seguem os quatro tipos definidos:

Servidor Início de jogo: 1 byte para a *flag* + 1 byte contendo o tamanho da palavra

Cliente Envia uma letra como tentativa de acerto: 1 byte para a *flag* + 1 byte contendo a letra palpite.

Servidor Envia a quantidade de ocorrências da letra na palavra desafio: 1 byte para a *flag* + 1 byte para a quantidade de ocorrências seguido de 1 byte contendo a posição das ocorrências.

Servidor Envia a *flag* de final de partida

Decisões sobre o protocolo

Para garantir o padrão de envio com cada informação sendo encapsulada em apenas um byte, foram utilizadas os tipos *char e unsigned char* implementados na forma de array, garantindo um buffer que foi devidamente tratado antes de cada comunicação.
As flags de controle foram convertidas para *unsigned char* e inseridas no buffer na ordem desejada pelo protocolo, bem como as demais informações como números inteiros e posições das letras na palavra desafio.
As letras por sua vez, foram convertidas e tem seu valor atrelado ao valor da tabela ASCII.

Implementação da comunicação

A comunicação foi definida de forma a respeitar o padrão imposto pelo código disponibilizado.
O servidor cria uma instancia e espera as conexões.

O cliente, espera a flag de início de rodada para entrar em um laço que termina a execução do programa bem como a conexão com o *socket* apenas ao fim da palavra, ou quando o usuário cancelar a execução.

Interrupção do servidor

A interrupção da execução do servidor pode ser alterada via flag dentro do código:

```
        close(csock);
        if(SINGLE_TURN) break;
    }
```

Através desta *flag*, o servidor pode ser configurado para suportar múltiplas partidas além de múltiplos jogares, porém cada jogador deve aguardar o final do jogo em andamento.
Com seu valor definido em 0, ao final do jogo, tanto cliente quanto servidor encerram suas execuções.

Definições de saída

Assim como o encerramento do servidor, a formatação bem como a quantidade de informação exibida no output pode ser definida através da *flag* **LOG** dentro do código.
Quando definida com valor 1, todas as mensagens de comunicação serão exibidas.
Esta opção está disponível em ambos os códigos, cliente e servidor.

```
matheus@note:~/Documentos/ufmg/redes/redes-computadores/trabalho-pratico1-forca$ ./server 51511
bound to IPv4 0.0.0.0 51511, waiting connections
[log] connection from IPv4 127.0.0.1 35030
[msg] Game started
[log] received 2 bytes
[log] char received: 112
[log] testing char: 'p' on word
[log] Index of occurrence of char: 'p' = 0
[log] Index of occurrence of char: 'p' = 19
[log] Occurrences of char: 'p' = 2
[log] size of occurence buffer 4 positions
[log] inserting 3 OCC_FLAG with 1 bytes
[log] inserting 2 occurrences with 1 bytes
[log] inserting index 0 with 1 bytes
[log] inserting index 19 with 1 bytes
[log] flag sent: 3
[log] sent 4 bytes
```

Compilar e executar

Seguindo as orientações da especificação, para compilar e executar o programa, basta executar o arquivo *makefile* que gera as saídas: *client* e *server*.

Comportamentos não implementados

Existe uma forma de vencer o jogo sem acertar exatamente todas as letras da palavra desafio.
Como não foi implementado um controle para definir quais letras já foram testadas, caso o usuário utilize mais de uma vez a mesma letra, o número de ocorrências será contabilizado também mais de uma vez.
Portanto, como o jogo leva em conta o número de letras corretas, basta acertar uma letra N vezes que o jogo se encerra.