Documentação do tp1 da Metaturma

Nome: Alessandro Azevedo Duarte

Matrícula: 2017072642

Foi utilizado o Código do professor Ítalo, disponibilizado no moodle, junto das suas vídeo aulas para estruturação da conexão com soquetes.

Temos dentro do projeto 3 arquivos em c, um cliente, um servidor e um common, também existe um Makefile para compilação dos programas que gera os executáveis.

No common existem algumas funções utilizadas tanto no server, quanto no client, como não foi adicionada nenhuma função nesse arquivo após as modificações do professor ítalo, não explicarei sobre as funções desse arquivo.

Acredito que não faz muito sentido explicar o que ele já explicou nas vídeo aulas, dessa forma, vou descrever abaixo as funções e lógicas que utilizei no código referentes as especificações do jogo da forca.

Como solicitado nas especificações para executar o servidor utilize o comando:

./servidor <porta>

E para executar o cliente execute:

./cliente <ip-servidor> <porta-servidor>

Tentei deixar o nome das funções o mais intuitivo possível e até mesmo o nome de algumas variáveis, antes das funções e lógicas do código existem comentários para descrever o que essa parte faz.

Programa servidor:

No início do programa utilizei uma variável global para a palavraChave que está no código como “ornitorrinco”, mas pode ser alterada e o código foi feito para se adaptar a cada palavra chave escolhida.

Após a conexão do cliente o jogo é iniciado com o envio do char msg, que tem em sua primeira posição o tipo de mensagem, que no caso é 1, como início do jogo e na segunda posição quantas letras essa palavra tem.

```
// Envia a mensagem de inicio e o tamanho da palavra após a conexão
        char msg[2];
        unsigned int lenKey = strlen(palavraChave);
        memset(msg, 0, BUFSZ);
        msg[0] = 1;
        msg[1] = lenKey;
        size_t cnt = send(csock, msg, strlen(msg) + 1, 0);
        if (cnt != strlen(msg) + 1) {
            logexit("send");
        }
```

A partir de agora a função do servidor é de receber os palpites e fazer a tratativa deles, dessa forma, criei um loop para ficar recebendo os palpites.

```
// Loop para o jogo
    while(1){
        char palpite[2];
        memset(palpite, 0, sizeof(palpite));
        size_t count = recv(csock, palpite, BUFSZ - 1, 0);
```

Se por algum motivo o jogador desiste do jogo (falha de conexão, ou um ctrl +c que sai da execução do programa no Linux) a mensagem abaixo é impressa no servidor e o mesmo fica aguardando por nova conexão.

```
if((int)count ==0){
printf("Palpite vazio e o Jogador desistiu\n");
    break;
}
```

Se o palpite é um palpite válido então os dados do cliente e o palpite são impressos na tela do servidor.

```
printf("Dados do jogador: %s\nTamanho da mensagem %d bytes\n O palpite foi> %s\n", caddrstr, (int)count, palpite);
```

Para verificar o palpite, foi criada uma função que retorna um vetor de inteiros, onde essa função recebe o palpite do jogador e a palavra para ser comparada, utilizando um loop(for) cada posição da palavra é comparada e em um vetor auxiliar são colocados a quantidade de posições que essa letra apareceu e nos próximos índices é colocado o valor da posição da palavra em que o palpite aparece.

```
// Variaveis e função para verificar se o palpite do jogador esta na palavra
        int aux[lenKey];
        int j = 1;
        aux[0] = 0;
        int *verificapalpite(char palp , char word[26]){
            for(int i = 0; i <strlen(word);i++){
                if (palp == word[i]){
                    aux[0] = aux[0]+1;
                    aux[j] = i;
                    j = j+1;
                }
            }
            return aux;
        }

        int *result = verificapalpite(palpite[0],palavraChave);
        char resultado[j+2];
        resultado[0] = 3;
        resultado[1] = result[0];
```

Posteriormente, foi criado um loop que colocada os valores das posições de forma alinhada no vetor resultado e incrementa a variável soma que será usada para verificar o fim do jogo.

```
        for( i=1;i <sizeof(resultado)-1;i++){
            resultado[i+1] = result[i];
            soma = soma + (int)result[i-1];

        }
```

Foi criada uma função que realiza uma soma recursiva do número passado.

```
        int somaRecursiva(int n){
            if (n ==1){
                return n;
            }else
                return n + somaRecursiva(n-1);
        }
```

Como mostrado acima a variável soma vai somando o seu próprio valor com o valor da posição em que o palpite aparece na palavra e como mostrado abaixo, foi utilizada a função acima para somar o tamanho da palavra de forma recursiva. Quando a variável soma é igual ao resultado da função somaRecursiva(), temos que o usuário conseguiu descobrir a palavra-chave, dessa forma, é enviada a mensagem de tipo 4 e é declarado o fim do jogo, após essa mensagem o servidor fica aguardando o próximo jogador.

```
if (soma == somaRecursiva(lenKey)){
    resultado[0] = 4;
     send(csock, resultado, sizeof(resultado) + 1, 0);
     printf("\nFIM DE JOGO\n\n");
     printf("Aguardando o proximo jogador...\n");
     break;
}
```

Se ainda faltam letras a serem descobertas é enviada a mensagem tipo 2, com o resultado do palpite.

```
memset(palpite, 0, sizeof(palpite));
size_t cntresult = send(csock, resultado, sizeof(resultado) + 1, 0);
if (cntresult != sizeof(resultado) + 1) {
    logexit("send");
}
```

Programa cliente:

Foi criada uma funcionalidade além da documentação que é a de verificar se o palpite realizado já foi digitado, para isso foram criadas essas duas funções que checam se o palpite já foi feito e a outra função que preenche um vetor de palpites já feitos.

```
//Função que checa se o palpite já foi digitado antes
bool checaPalpite(char palpite, char *palpitesFeitos){

    for(int i = 0; i<sizeof(palpitesFeitos);i++){
        if (palpite == palpitesFeitos[i]){
            return true;
        }
    }
    return false;
}
```

```
  void preenchePalpites(char newpalpite, int cnt){
      palpitesFeitos[cnt]= newpalpite;
  }
```

Da mesma forma como no servidor foi criada um loop para recebimento das mensagens do servidor. O vetor palpite é o vetor que recebe os palpites inseridos pelo usuário.

```
    while(1){
        char buf[8];
        memset(buf, 0, BUFSZ);

// Recebe a mensagem de inicio após a conexão e esvazia o palpite
        size_t cnt1 = recv(s, buf, BUFSZ, 0);
        char palpite[2];
        palpite[1] = ' ';
```

Na posição 0 do vetor buf recebemos o tipo de cada mensagem, assim, eu criei condicionais para cada mensagem.

A mensagem 1 é a de início do jogo e após o início, utilizei a função scanf para receber cada palpite e após receber esses palpites fui armazenando-os em um vetor, utilizei o contador countPalpite para ser o índice do vetor de palpites preenchidos. Após receber o palpite esse palpite é enviado para o servidor.

```
// Opção para inicio do jogo
    if (buf[0]==1){
        printf("Inicio do jogo\n\n");
        printf("A palavra a ser respondida tem %d letras\n\n", buf[1]);

        memset(palpite, 0, sizeof(palpite));
        printf("Atencão!!\n\nDigite apenas uma letra\n\nSe mais de uma letra for digitada será considerada apenas a primeira\n\n");
        printf("Digite seu palpite e pressione enter> ");
        scanf(" %c", &palpite[1]);

        preenchePalpites(palpite[1],countPalpite);
        countPalpite++;
        size_t count = send(s, &palpite[1], sizeof(palpite[1]), 0);
        if (count != sizeof(palpite[1])) {
            logexit("send");
        }
    }
```

A mensagem de tipo 3 é a de resposta do palpite, dessa forma é informado ao jogador quantas vezes seu palpite apareceu na palavra e mensagens de parabéns ou não por ter acertado o palpite, após essa primeira verificação o usuário tem de continuar o jogo digitando os próximos palpites.

```
// Opção após realização do primeiro palpite e realização dos próximos
    if (buf[0]==3){
        printf("\n*********Resposta do seu palpite********\n");
        printf("Quantas vezes seu palpite aparece na palavra> %d\n", buf[1]);
        if(buf[1]>0){
            printf("\nParabéns!!!\n\nSeu palpite está presente na palavra\n\nNas posicões\n\n");
        }
        if(buf[1]==0){
            printf("\nAhh, que pena!\n\nSeu palpite não está presente na palavra, mas tente novamente!\n\n");
        }
        for( int i = 0; i<(int)cnt1-4;i++){
            printf("Posição %d \n",buf[i+2]+1);
        }

        printf("Digite seu novo palpite> ");
        scanf(" %c", &palpite[1]);
```

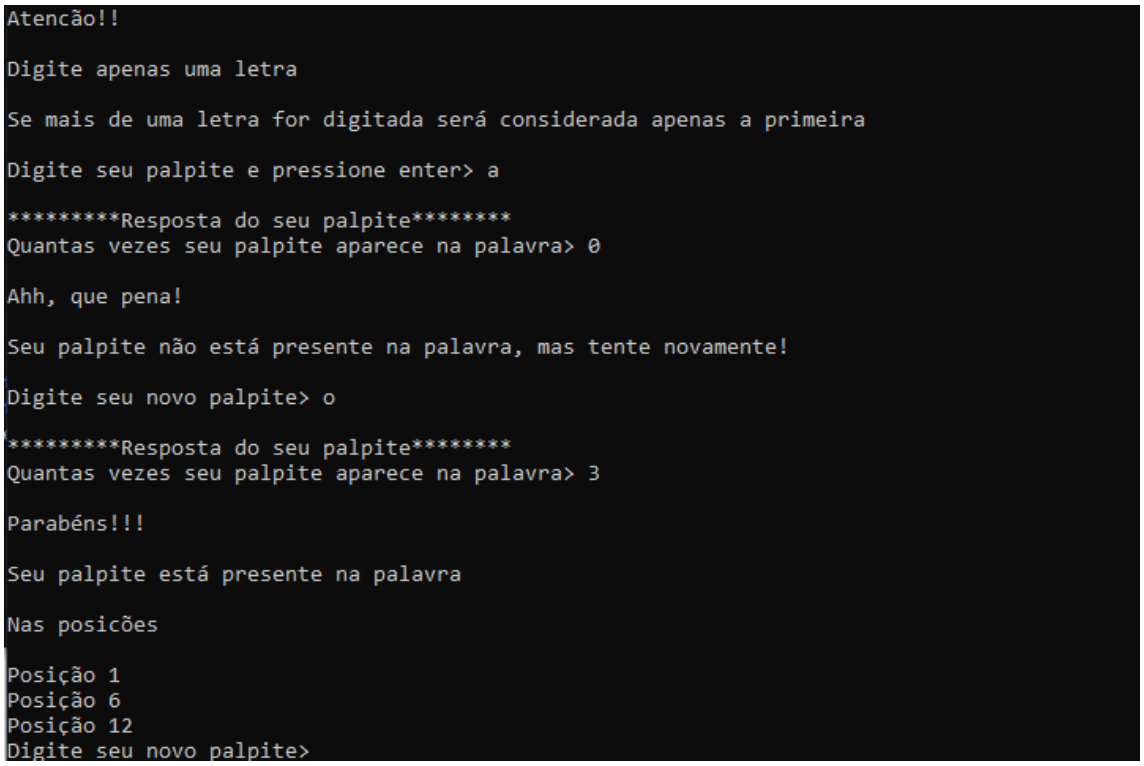
Por meio desse loop apresentado abaixo eu verifico se o palpite já foi realizado, se já foi, um novo palpite é solicitado, se não o vetor de palpites é incrementado e o palpite é enviado ao servidor.

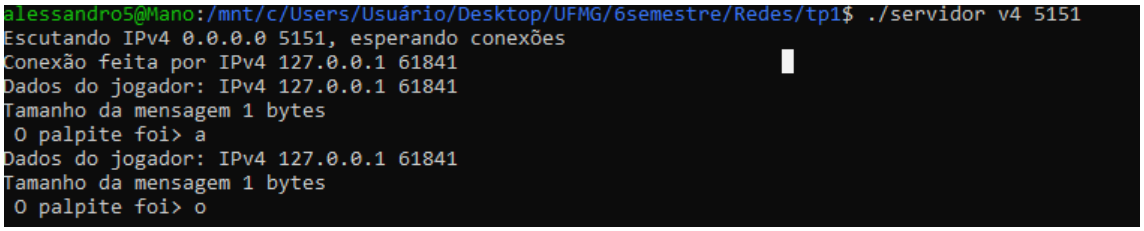
```
while(checaPalpite(palpite[1],palpitesFeitos)){
    printf("\n----------Voce ja enviou essa letra----------\n");
    printf("Digite outro palpite> ");
    scanf(" %c", &palpite[1]);
}
preenchePalpites(palpite[1],countPalpite);
countPalpite++;
size_t count = send(s, &palpite[1], sizeof(palpite[1]), 0);
if (count != sizeof(palpite[1])) {
    logexit("send");
}
```

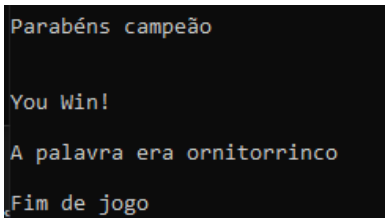
A mensagem tipo 4 é a enviada após o usuário acertar todas as letras da palavra e o usuário recebe mensagens de felicitações e a palavra Chave é mostrada pala o cliente e a conexão do cliente é finalizada.

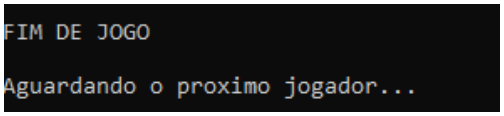
```
//Opção após acertar todas as letras da palavra
    if (buf[0]==4){
        printf("\nParabéns campeão\n\n");
        printf("\nYou Win!\n\n");
        printf("A palavra era %s\n\n",palavraChave);
        printf("Fim de jogo\n");

        break;
        close(s);
    }
    }

    exit(EXIT_SUCCESS);
}
```

Optei por fazer uma interface um pouco interativa com o usuário para que de acordo com cada palpite a resposta fosse mais animada para que o jogador se sinta feliz para continuar jogando, como mostram as fotos abaixo:









Um possível incremento no código é utilizar as funções srand, e rand para gerar palavras aleatórias dentre de um vetor de palavras definidas, mas como na especificação a palavra seria definida previamente optei por defini-la no início do programa.

Gostei bastante de fazer esse trabalho e consegui visualizar bons conceitos da matéria, no tp0 acabei fazendo as coisas um pouco em cima da hora e tive alguns problemas pessoais, mas nesse trabalho fui me dedicando aos poucos e estou feliz com o resultado final, espero que quem esteja avaliando também goste!