Decisões da Arquitetura do Código

Arquitetura

**Estrutura de Dados**

Defines importantes:

**MAXLINE = 4096**

**MAXDATASIZE = 100**

**- Matriz palavras[MAXLINE][MAXDATASIZE]:** Responsável por manter o dicionario no programa sem demandar leitura de um arquivo externo o tempo todo.

**- Vetor alfabeto[26]:** contém em cada posição as 26 letras do alfabeto.

**- Vetor usadas[26]:** 26 flags referentes a cada uma das 26 letras se já foram usadas para advinhar a palavra.

**- Vetor encontradas[tamanho da palavra atual]:** Responsavel por manter uma flag associada a cada letra da palavra, quais delas foram advinhadas e podem ser exibidas no print atual.

**- Vetor buf[MAXLINE]:** Responsável por conter as mensagem que vão do cliente para o servidor e servidor para cliente.

**Funções**

No servidor em ordem:

**- getIp:** Responsavel por obter o IP do cliente.

**- printArquivo:** Pega o dicionario e adiciona na nossa estrutura de dados ou imprime README a depender da flag fornecida.

**- initVetor:** Auxiliar para zerar o vetor utilizado

**- verificaVetor:** Verifica se todas as palavras do dicionario já foram usadas.

**- escolhePalavra:** Escolhe aleatóriamente uma nova palavra para uma nova partida.

**- menu:** Impressão do menu que apresenta as 3 opcoes de jogo

**- gameover:** Funcao para quando a vida acaba e voce perde

**- procuraAlfabeto:** Mantem o controle de que letras ja foram usadas ou nao

**- procuraLetra:** Faz o controle se a letra que foi dada pelo cliente tem na palavra, se tiver revela. Se não tiver perde uma vida.

**- completouPalavra:** Verifica se a palavra foi completamente desvendada.

**- partidaSimples:** A funcao responsavel por executar todo o modo de jogo da opção 1.

**- serCarrasco:** A funcao responsavel por executar todo o modo de jogo da opção 2.

**- multiplayer:** A funcao responsavel por executar todo o modo de jogo da opção 3.

**- main:** Responsavel por fazer a conexao, controlar a partida e chamar os diferentes modos de jogos a depender do que digitou no menu.

No cliente temos apenas o main, responsável pela conexão com o servidor e o envio das mensagens para ele. O servidor com as mensagens enviadas pelo cliente gerencia todo o resto.

Implementação das Regras

Se o cliente digitou uma letra, vamos procurar no alfabeto

if(strlen(buf) == 1 || strlen(buf) == 0){

i = procuraAlfabeto(alfabeto, buf, usadas);

e na funcao procuraAlfabeto verificamos se já foi usada ou não, e se é invalida ou não.

int procuraAlfabeto(char alfabeto[26], char \*letra, int usadas[26]){

for(int i = 0; i < 26; i++){

if(alfabeto[i] == \*letra){

if(!usadas[i]){

usadas[i] = 1;

return 1; //letra ainda nao usada

} else {

return 2; //letra ja usada

}

}

}

return 0; //letra invalida

}

Se a seção anterior retornou 1, indicando letra não usada ainda. Ele procura a existência dessa letra na palavra para setar as flags de que letras foram descobertas. Se todas as letras já foram descobertas decreta a vitória.

Se retornou 2, ela printa que a letra já foi utilizada já que repetição é avisada sem dano ao jogador, da mesma forma que o aviso e letra inválida.

if(i == 1){

procuraLetra(atual, buf, len-1, &vidas, encontradas);

if((vidas) == 0){

gameover(len, total\_vidas, atual);

return;

}

if(completouPalavra(encontradas, len-1)){

palavras[atual][len -1] = '\0';

printf("Você adivinhou a palavra '%s'! Parabéns!\n", palavras[atual]);

return;

}

} else if(i == 2){

printf("A letra %s já foi utilizada\n\n", buf);

} else{

printf("Letra inválida\n\n");

}

Revelar todas as letras igual a letra que advinhei.

void procuraLetra(int atual, char \*letra, int len, int \*vidas, int \*encontradas){

int achou = 0;

for(int i = 0; i < len; i++){

if(palavras[atual][i] == \*letra){

achou = 1;

encontradas[i] = 1;

}

}

if(!achou){

printf("A palavra não tem nenhuma letra '%c'.\n", \*letra);

(\*vidas)--;

printf("Você agora possui %d vidas.\n\n", \*vidas);

} else {

printf("\nPALAVRA:");

for(int i = 0; i < len; i++){

if(encontradas[i]){

printf(" %c", palavras[atual][i]);

} else {

printf(" \_");

}

}

printf("\n\n");

}

}

Se todas as letras foram reveladas, decreta a vitória

int completouPalavra(int \*encontradas, int len){

for(int i = 0; i < len; i++){

if(!encontradas[i]){

return 0;

}

}

return 1;

}

Caso o jogador digite uma palavra, qualquer string maior que 1, as opções são:

A palavra é “-h” e ele pediu pela ajuda, que é o README.

A palavra é “exit” e o processo é encerrado e ele tem a derrota decretada, a palavra revelada antes de ser desconectado.

A palavra é a palavra atual, é decretada a vitória.

A palavra não é a palavra atual, nem “exit”, nem “-h”, é decretado a derrota sem o encerramento do processo.

} else { // se cliente digitou uma palavra

palavras[atual][len -1] = '\0';

if(strcmp(buf, "-h") == 0){

printArquivo("README.txt", 1);

} if(strcmp(buf, "exit") == 0){

printf("Voce desconectou antes do final da partida!\n");

printf("A palavra correta era %s, você perdeu!\n\n", palavras[atual]);

close(connfd); /\* done with this client \*/

exit(0);

} else {

if(strcmp(palavras[atual], buf) == 0){

printf("Você adivinhou a palavra!\n\n");

return;

} else {

printf("A palavra correta era %s, você perdeu!\n\n", palavras[atual]);

return;

}

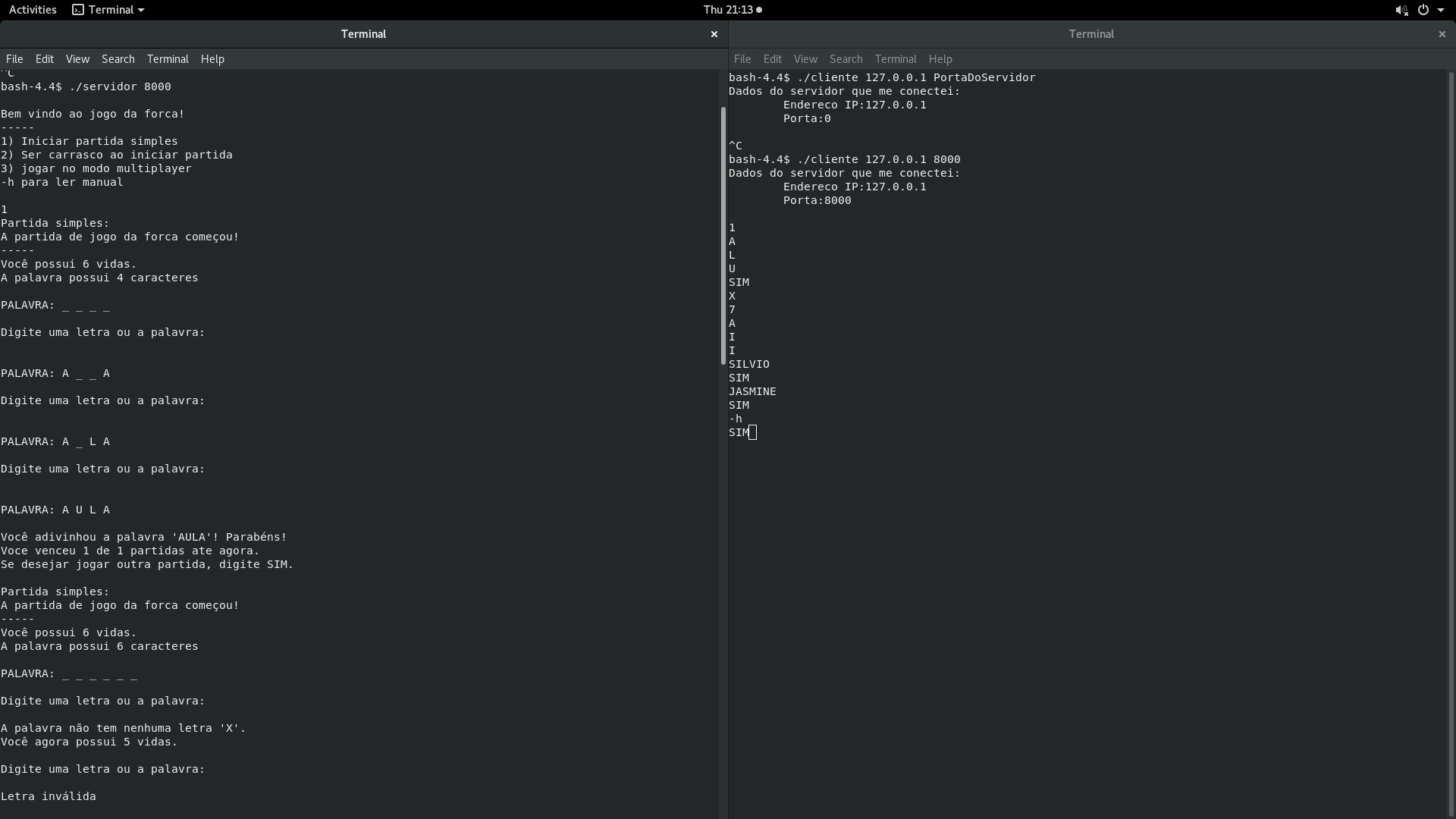
}

}

}

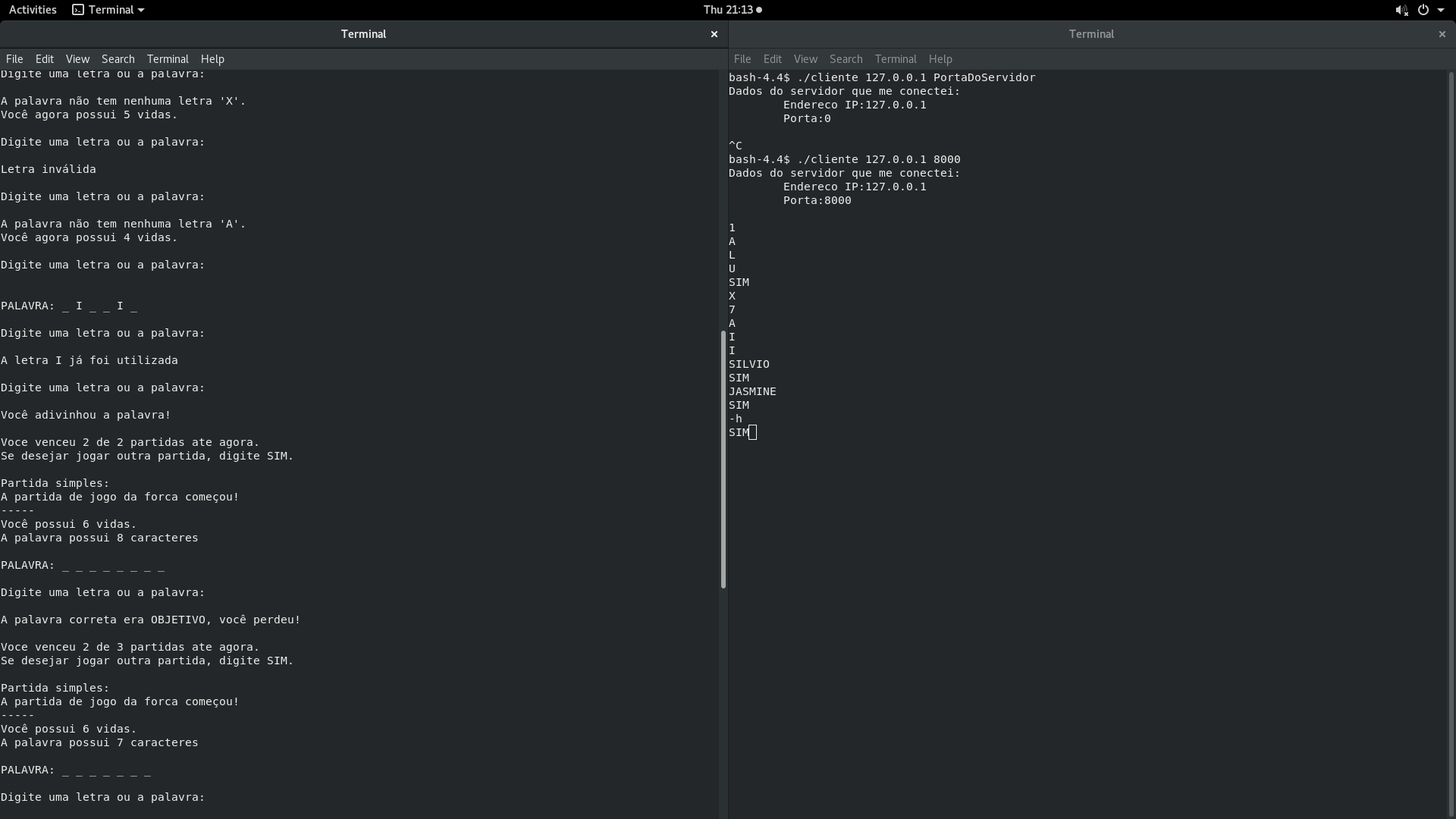
}

Prints

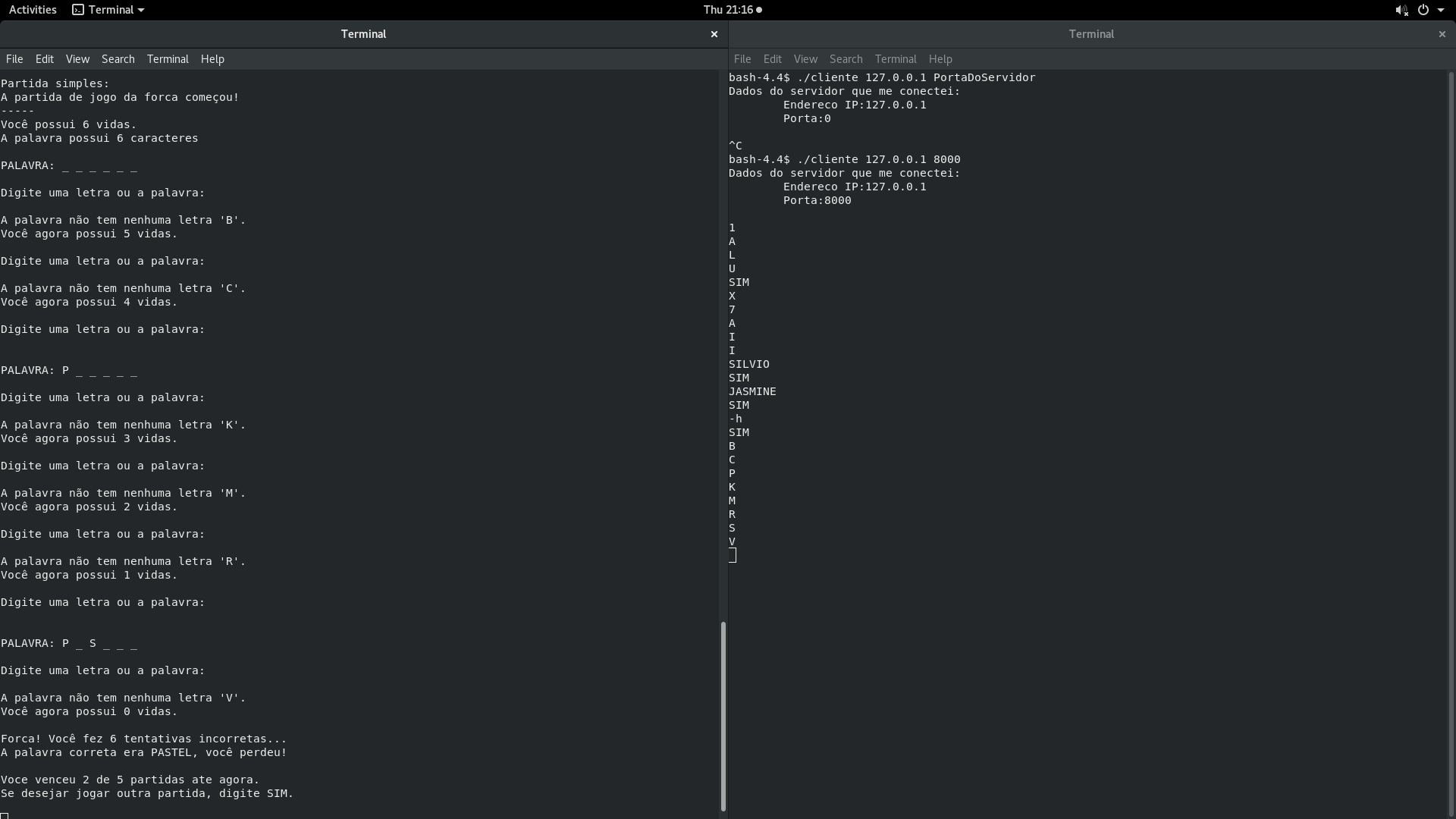


Print 1: Iniciamos o jogo, e na sequência iniciamos a partida simples.

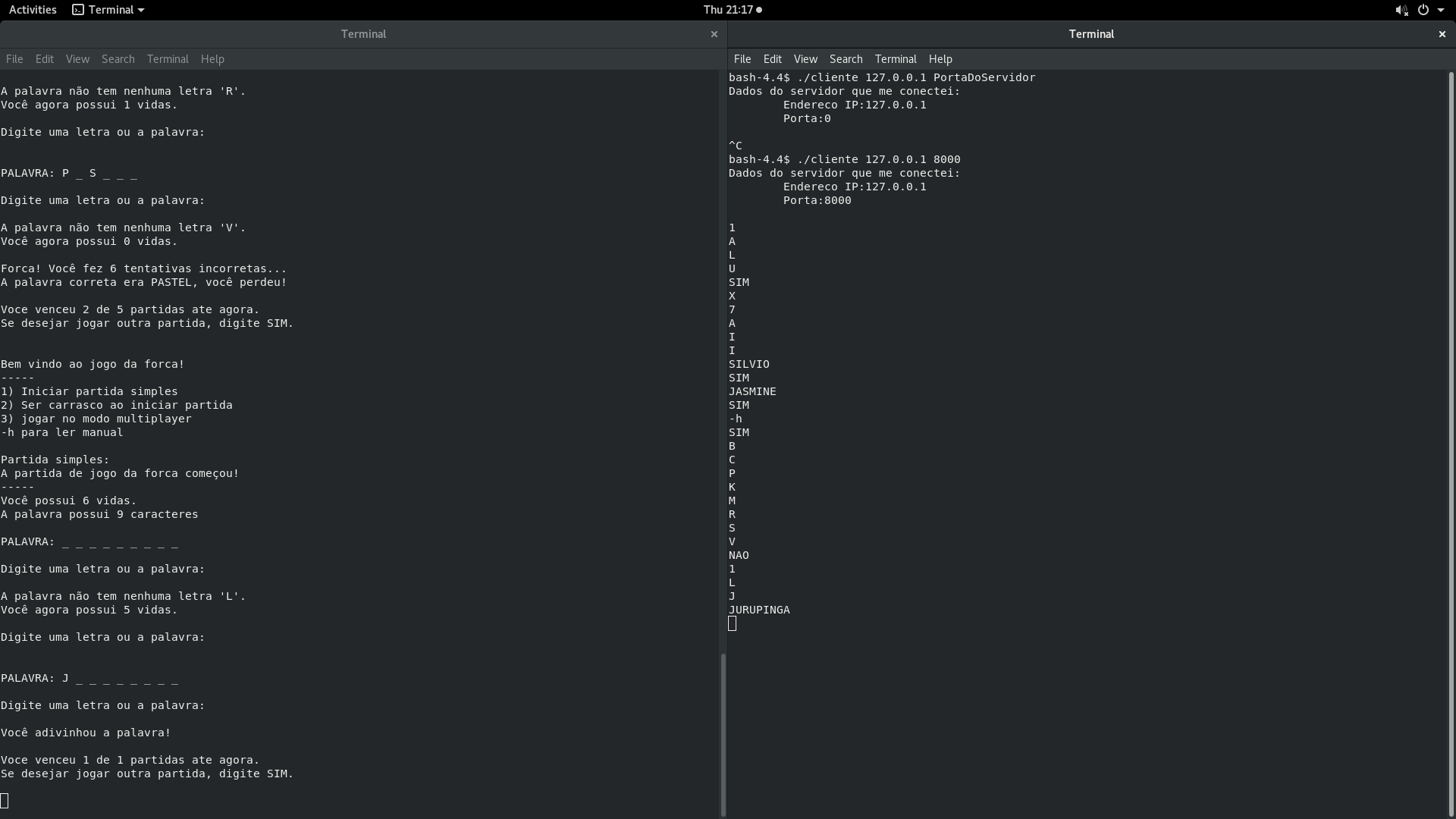
Advinhamos a palavra aula letra por letra. Depois com o “SIM” reiniciamos a partida simples. Onde erramos uma letra ao digitar X.



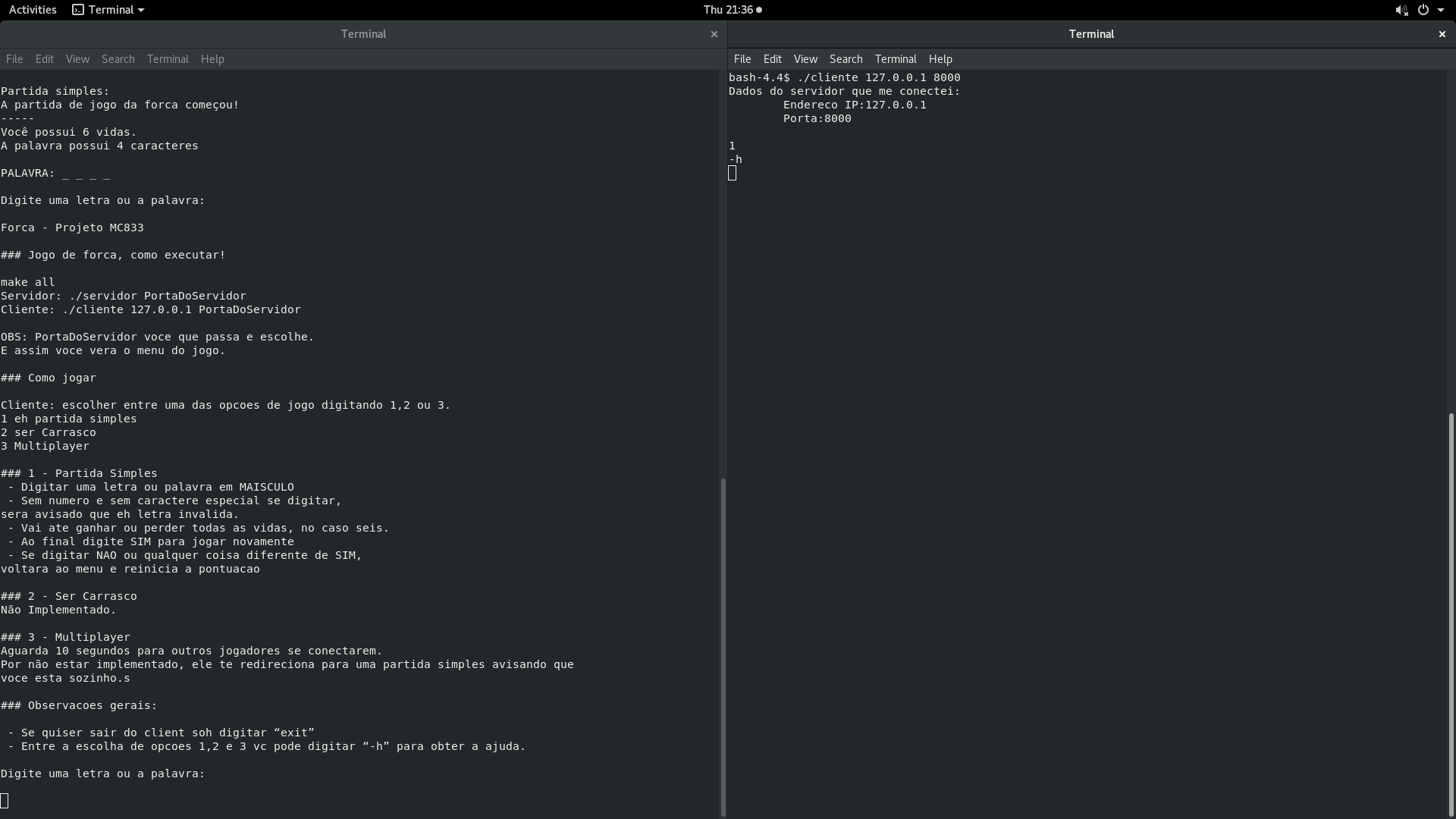
Print 2: Inserimos 7 uma letra invalida. Inserimos a letra I duas vezes para mostrar que repetição não conta mas não há penalidade. Com “Silvio” podemos advinhar a palavra toda de uma vez. Reiniciamos com SIM novamente. Com “jasmine” advinhamos a palavra errada. Com sim reiniciamos.



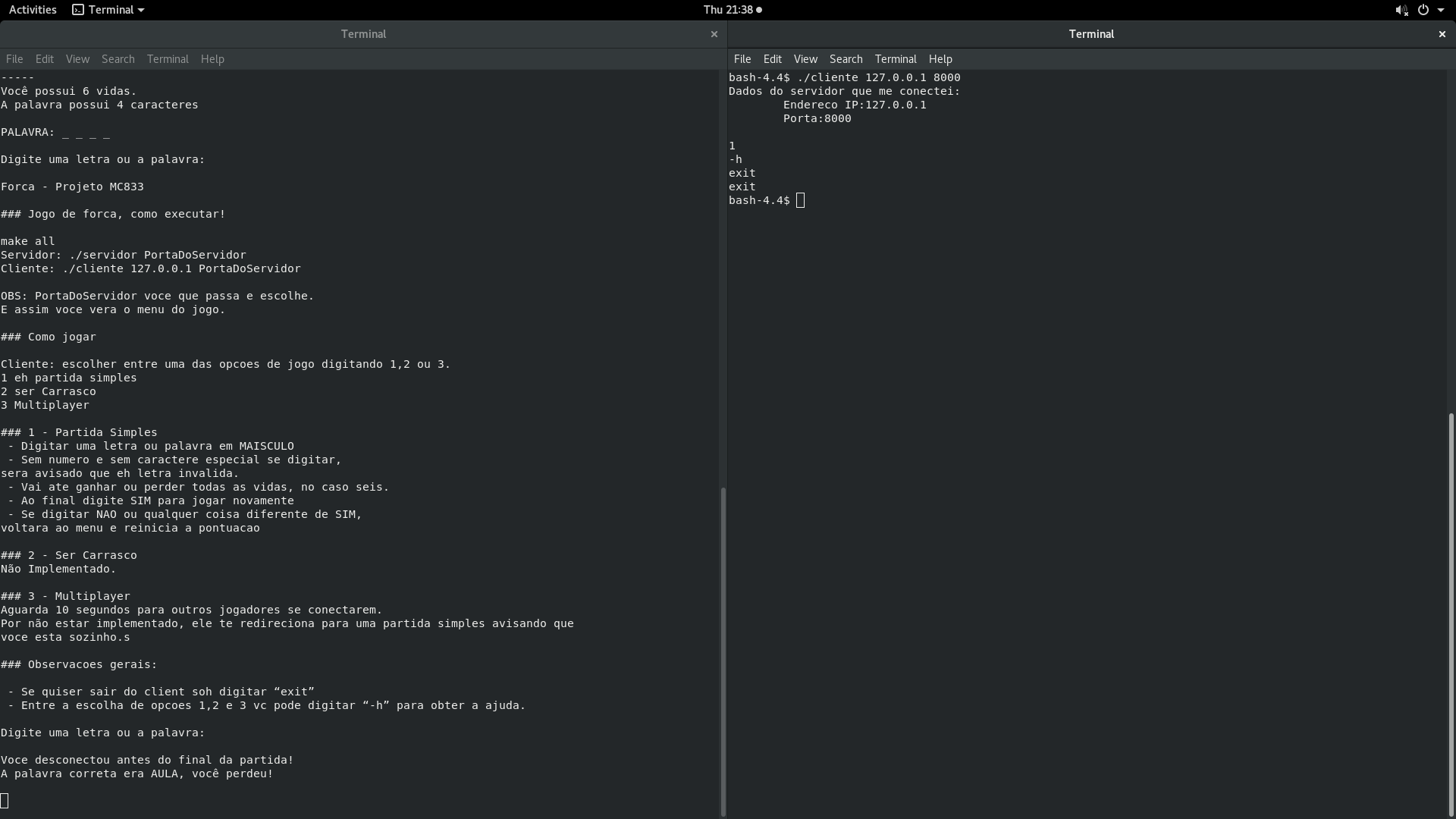
Print3: Chutando 6 letras erradas da palavra pastel perdemos uma vida de cada vez. Fazemos isso ao inserir as letras BCPKMRSV



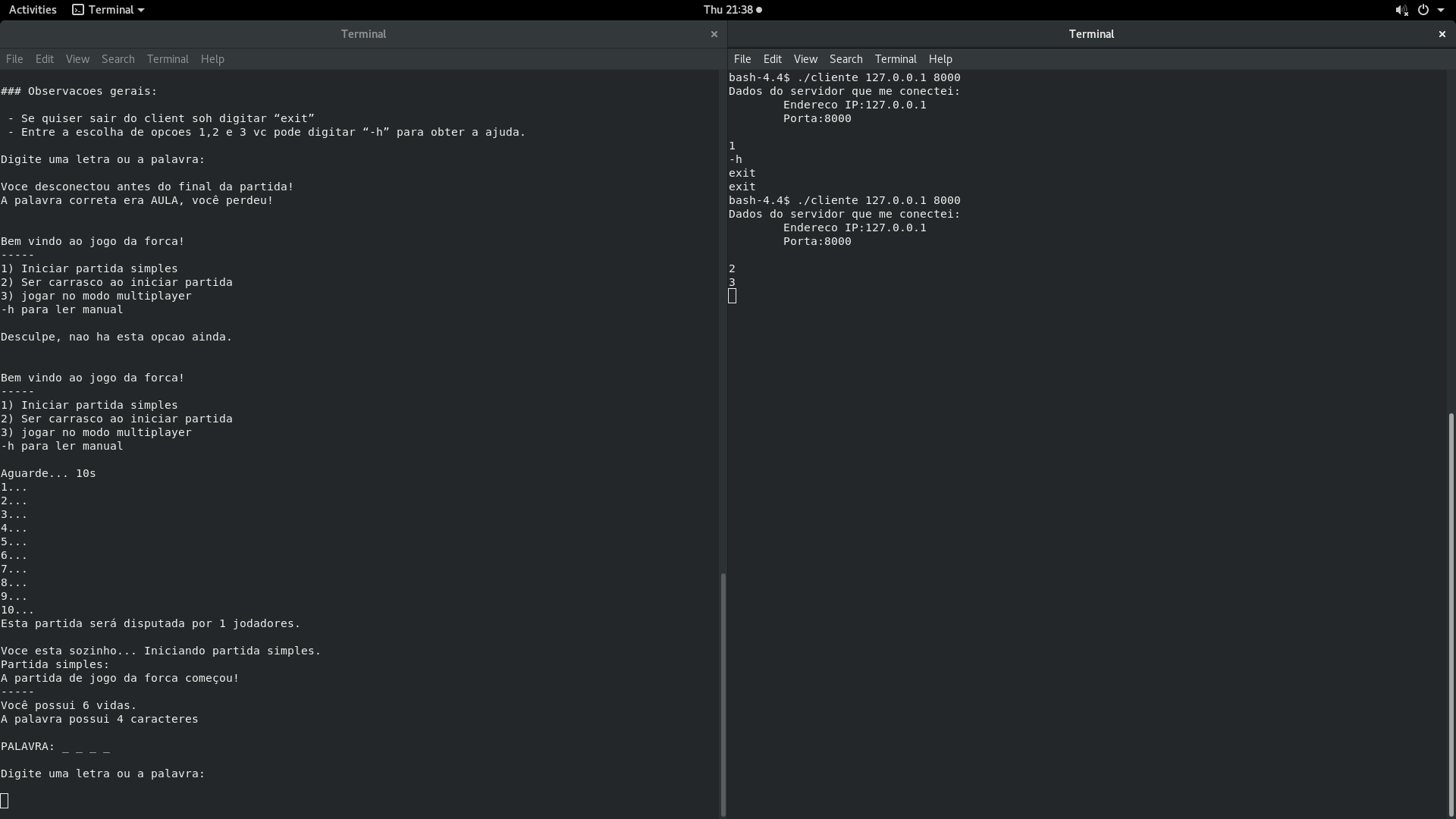
Print 4: Ao digitar NAO retornamos ao menu, e digitando 1 caimos numa partida simples. Onde advinhamos corretamente mas dessa vez chutando a palavra e não letra por letra.



Print 5: Digitando -h aparece o README



Print 6: Ao digitar “exit” é considerada a desistência da partida. O cliente imprime o próprio exit também conforme um dos nossos laboratórios.



Print 7: Mostrando as opções 2 e 3 do menu.