Projeto Final

Hangman Game

Laboratório de Computadores
2019/2020

Turma 2 - Grupo 10

André Assunção - up201806140
Marta Lobo - up201604530

Índice

1. Secção I: Instruções de utilização

1.1. Menu principal

Ao iniciar o jogo, aparece o menu inicial com as seguintes opções:

- Jogar, permite ao utilizador iniciar um novo jogo.
- Highscores, onde aparecem as 10 melhores pontuações.
- Opções, redireciona para um menu de personalização do jogo.
- Sair, sai do programa.

Figura 1 - Menu Principal

O menu principal permite usar o rato para selecionar as opções pretendidas.

1.2. Jogo

Quando o utilizador escolhe a opção ‘JOGAR’ é direcionado para uma adaptação do tradicional jogo da Forca. O objetivo é um jogador acertar a palavra proposta pelo computador, tendo como dica o número de letras. A cada letra errada, é desenhada uma parte do corpo do ‘enforcado’.

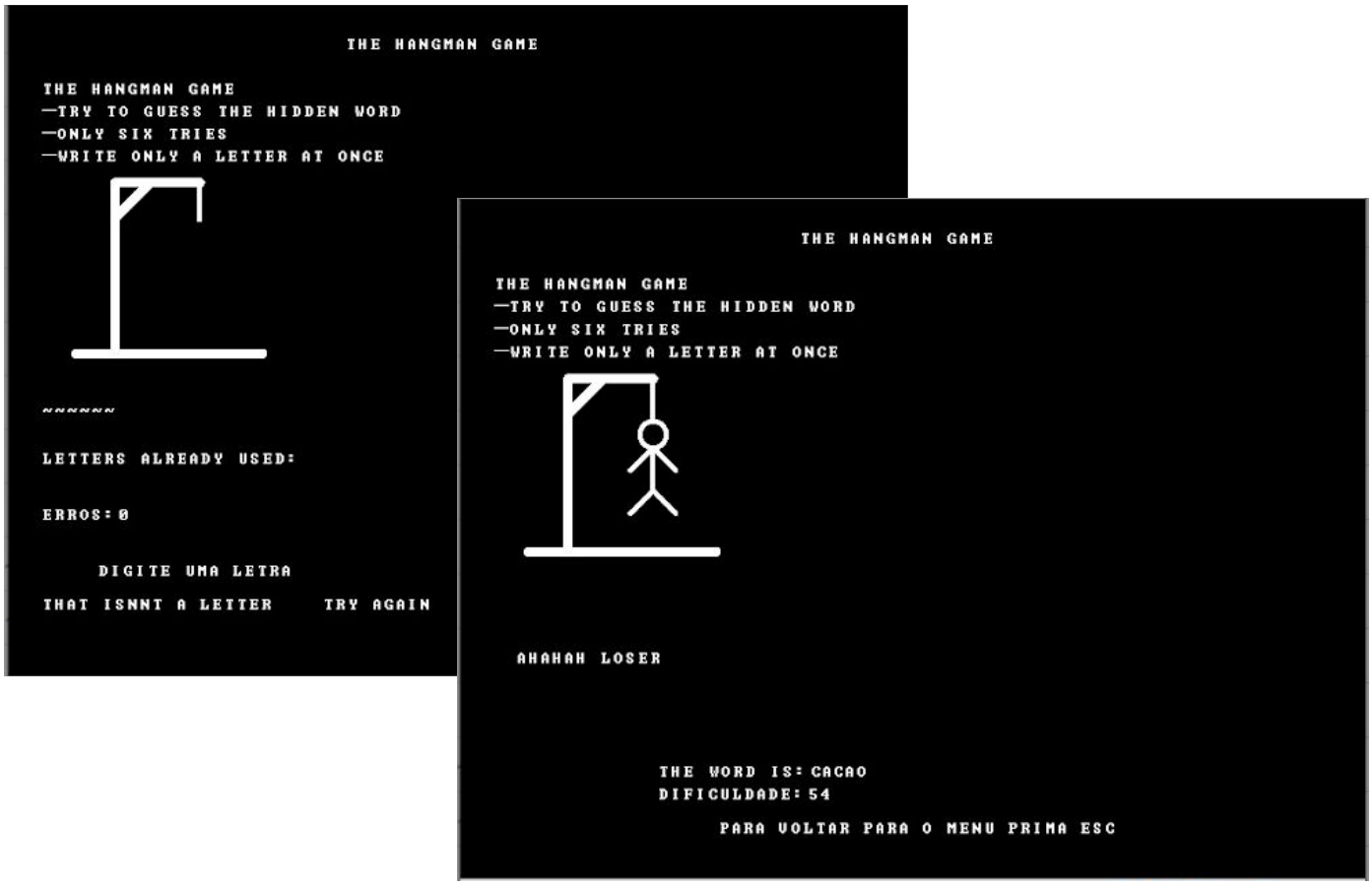


Figura 2 - The Hangman Game

O jogo pode terminar de três formas:

- se o jogador acertar a palavra;
- se esgotar o número de tentativas e, por consequência, preencher por completo o corpo do ‘enforcado’;
- se não fizer a sua jogada dentro do tempo definido (60 segundos).

Para adivinhar as letras da palavra, o utilizador deve recorrer ao teclado.

1.3. Menu de Opções

Neste menu, o jogador pode escolher a dificuldade pretendida, havendo 4 opções - Fácil, Médio, Difícil e Extreme; poderá também personalizar as cores do jogo.

Ao iniciar o programa, o jogo terá, por predefinição, o menu a preto, o jogo a preto e dificuldade média.

Assim que o utilizador estiver satisfeito com as suas escolhas, ao clicar em Sair, será redirecionado para o menu principal onde poderá iniciar o jogo.

Figura 3 - Menu de opções

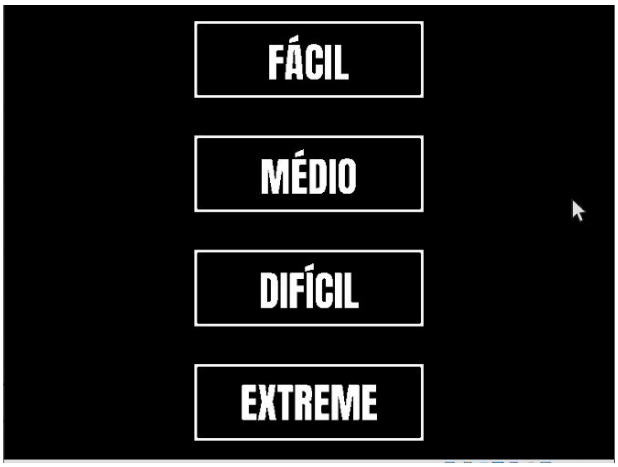


Figura 4 - Escolha de dificuldade

Figura 5 - Escolha de cores

Figura 6 - Nova cor escolhida

1.4. Highscores

Nesta opção do menu principal são mostradas as melhores pontuações dos jogadores seguidas da hora e data em que foram conquistadas.

Figura 7 - Highscores

2. Secção II: Estado do Projeto

2.1. Dispositivos Usados

A seguinte tabela contém a informação acerca dos periféricos utilizados, a sua utilidade e se usam interrupções ou não.

Os "ciclos de interrupções" (função onde são chamados os *handlers* de cada periférico e a função *driver_receive()*) localizam-se no módulo *proj* e no módulo *vídeo.*

Dispositivo	Função	Interrupções
Timer	Determinar o tempo de jogada	Sim
Teclado	Inserir as letras	Sim
Rato	Navegação nos menus	Sim
Placa Gráfica	*Display* de imagens	Não
RTC	Leitura da data e hora	Não

Tabela 1 - Dispositivos Utilizados

2.2. Timer

A principal função do timer é a determinação do tempo de jogada.

Para o controlo do tempo foram utilizadas as interrupções do timer (recorremos à sua frequência base, 60 interrupções por segundo). Foram implementadas funções para subscrição de interrupções (ao longo do laboratório 2), sendo essas funções *timer_subscribe_int()* e *timer_unsubscribe_int()*.

O módulo *proj*, na função *(proj_main_loop)(),* foi onde foram desenvolvidas as funcionalidades deste periférico.

2.3. Teclado

Este dispositivo é usado na lógica do jogo. Durante o jogo, serve para o utilizador ir adivinhando as letras contidas na palavra. É utilizado como text input, mas o text input é já de si o controlo do jogo, dada a sua natureza linguística.

São utilizadas interrupções para a verificação das teclas premidas. Foram implementadas funções para subscrição de interrupções (ao longo do laboratório 3), sendo essas funções *kbd_subscribe_int()* e *kbd_unsubscribe_int()* (contidas em *keyboard.c*). Igualmente importante

para o funcionamento deste dispositivo, contida no mesmo ficheiro, é a função *kbc_ih()*.

O módulo *proj*, na função *(proj_main_loop)(),* foi onde foram desenvolvidas as funcionalidades deste periférico.

Nos highscores e no fim de cada jogada, se o utilizador pressionar a tecla 'ESC', é direcionado para o menu principal.

2.4. Rato

É a forma de navegação em todos os menus, a partir da posição do cursor. Os menus têm vários botões que o utilizador pode selecionar com o rato, sendo utilizados tanto os botões como o movimento.

Funciona por interrupções, logo, sempre que há um clique no botão é gerada uma interrupção que é devidamente tratada pelo handler. No ficheiro *mouse.c* estão inseridas várias funções fulcrais para o correto funcionamento deste dispositivo, tais como *packet_processing(), kbc_write(), mouse_ih()* e as funções para subscrição de interrupções feitas no laboratório 4, *subscribe_mouse_int()* e *unsubscribe_mouse_int()*.

O módulo *menu*, na função *(menu_init)(),* foi onde foram desenvolvidas as funcionalidades deste periférico.

2.5. Placa Gráfica

Este dispositivo é utilizado para fazer o *display* das imagens usadas no jogo. O modo vídeo que decidimos usar é 0x115 (800x600, direct color, 24 Bits Per Pixel (8:8:8)).

Ao longo de todo o programa é utilizado double buffering de forma a aumentar a eficiência do programa, nomeadamente na escrita de palavras em modo gráfico.

No menu inicial é também implementado collision detection, de forma a utilizar o cursor elegantemente, sem uma "caixa" à volta, pois no modo de vídeo utilizado não existe transparência.

Numa análise menos linear, o sprite da forca pode ser considerado, de certa forma, um animated

sprite, pois vai alterando a sua forma à medida que o programa prossegue.

Este periférico é desenvolvido no módulo *video.*

2.6. Real Time Clock (RTC)

Este dispositivo é utilizado para ler a data e hora a que foram obtidas as pontuações dos utilizadores.

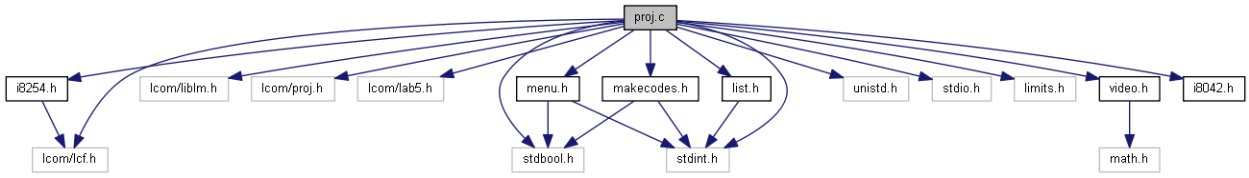
Este periférico é desenvolvido no módulo *proj,* na função *wait_valid_rtc()*.

3. Secção III: Organização e Estrutura do Código

No desenvolvimento do projeto, dividimos o código em diversos módulos, de forma a tornar o código mais estruturado e legível. Os diagramas de chamada de funções foram gerados pelo Doxygen, pelo que estão contidos na documentação gerada todos os gráficos de chamadas de funções. No relatório não estão presentes os gráficos dos módulos *video* e *menu* por ocuparem um espaço muito elevado.

Proj

Este módulo é fundamental para o funcionamento do programa uma vez que inclui funções responsáveis por vários periféricos.



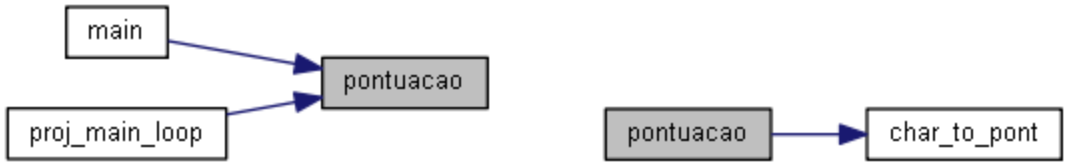
Algumas das funções mais importantes neste módulo são:

- ***void wait_valid_rtc*** *(uint32_t* hour, uint32_t *min, uint32_t *seg, uint32_t *day, uint32_t *month, uint32_t *year)*

  É onde ocorre o desenvolvimento do código responsável pelo funcionamento do Real Time Clock (RTC). Este periférico tem como função a apresentação da data e hora actuais. Quando um utilizador termina a jogada, a sua pontuação é guardada com as respetivas data e horas. Funciona por *polling*.

- ***int pontuação*** *(char *pal)*

  Algoritmo para determinar a pontuação de um utilizador após a sua jogada. A pontuação está relacionada com a complexidade das palavras adivinhadas, podendo esta complexidade pertencer a 4 grupos distintos: fácil, médio, difícil ou extreme.

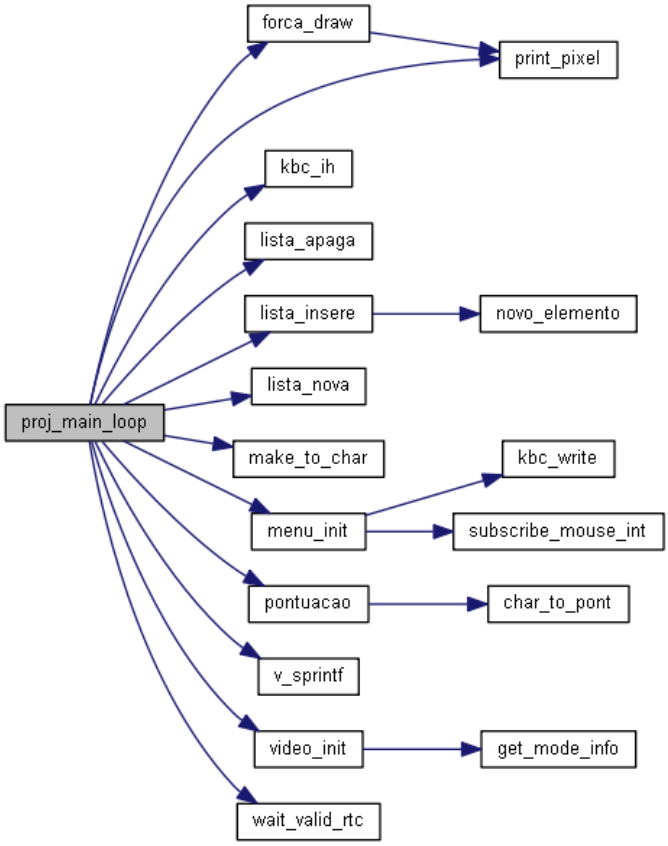


- ***int (proj_main_loop)*** *(int argc, char *argv[])*

  É onde se inicializam serviços importantes tais como chamada à função *video_init*.

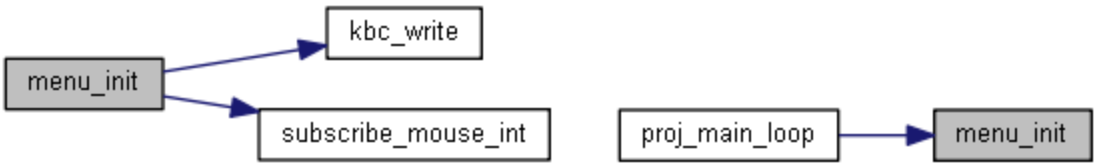
  Toda a lógica geral do jogo é desenvolvida nesta função. É utilizado um ficheiro *.txt ("/home/lcom/labs/proj/words.txt")* para a escolha da palavra a adivinhar pelo jogador.

  Nesta função estão localizadas as interrupções do teclado realizadas para o reconhecimento das letras inseridas pelo utilizador.



Menu

Este módulo é responsável pelos menus. A função *int menu_init(int *option)* é onde se desenvolvem funcionalidades do rato no menu principal e display dos botões de opções.



A função *cores_menu()* é responsável pelo menu de personalização das cores do jogo. Recorre à função *vg_draw_rectangle(),* desenvolvida no laboratório 5, para fazer o display das 4 opções de

cores. A função *dificuldade_menu()* é responsável pelo menu de personalização da dificuldade do jogo.

Já a função *opções_menu()* tem em vista o display destas últimas opções de personalização do jogo.

Todos os menus são manuseados com o rato, pelo que o desenvolvimento de funcionalidades desse periférico está presente em todas as funções anteriores.

Os menus são, no fundo, uma state machine, em que cada menu é um estado.

Video

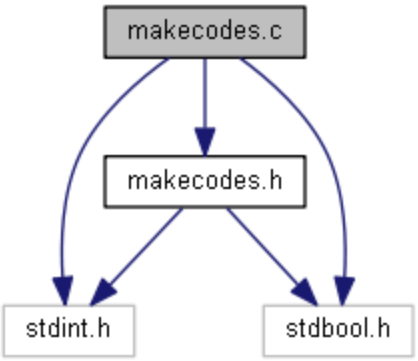
Neste módulo encontram-se as funções desenvolvidas durante o laboratório 5 relativas à placa gráfica e foram adicionadas outras funções.

A função *forca_draw()* é responsável pelo *display* das imagens do "enforcado". Foi novamente utilizada a função *xpm_load()*, com o objetivo de ler os pixmaps correspondentes a cada fase da forca ao longo das tentativas falhadas do utilizador.

Na função *v_sprintf()* foram utilizadas imagens do tipo *xpm* para cada letra, número e caracteres utilizados. Recorremos a função *xpm_load()* providenciada no laboratório 5; esta função lê um pixmap semelhante ao xpm definido na variável "map" e retorna o endereço da memória alocada para onde a imagem foi lida no formato especificado. Esta função "transforma" uma string no respetivo equivalente gráfico.

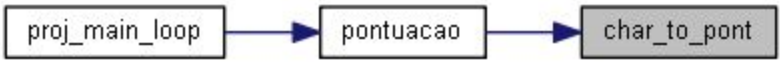
Makecodes

Deste módulo fazem parte duas funções, *make_to_char()* e *char_to_pont()*.



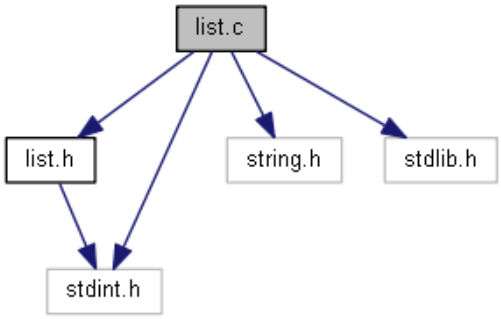
A primeira é utilizada para associar a cada *makecode* recebido do teclado a letra respetiva.

Já a segunda função é utilizada na determinação da pontuação de um jogador, associando a cada letra uma pontuação própria (baseada no clássico jogo de tabuleiro "Scrabble").



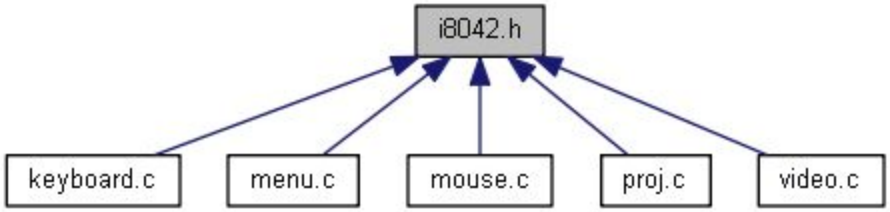
List

Este módulo tem algumas funções relacionadas com listas, estrutura de dados utilizada para armazenar e alterar os highscores, com a dificuldade e a data em que foram obtidos (funcionalidade do RTC).



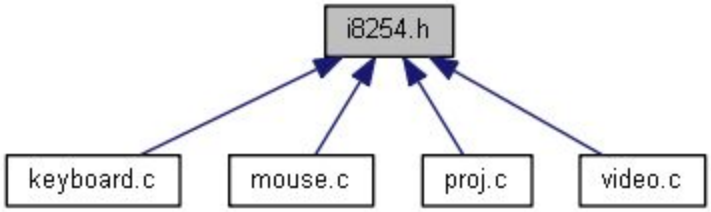
I8042

Neste módulo encontram-se constantes para manipulação do keyboard e também do rato. Importado do código das aulas práticas dos laboratórios 3,4 e 5.



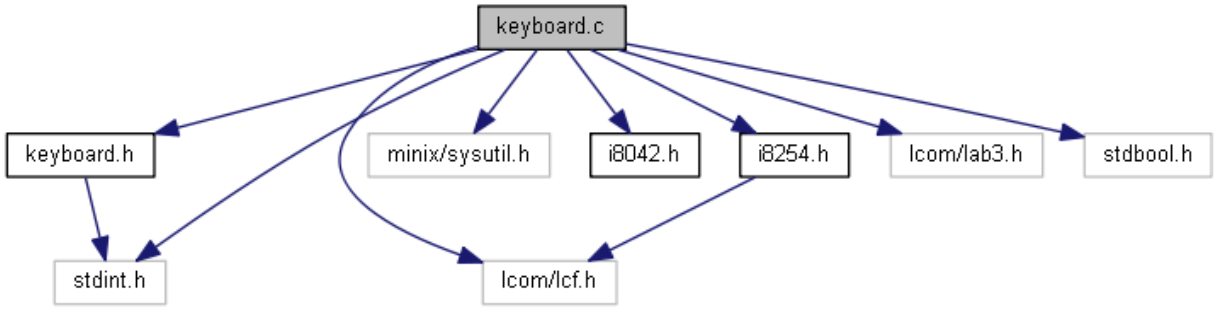
I8254

Neste módulo encontram-se todas constantes para manipulação do timer. Importado do código das aulas práticas do laboratório 2.



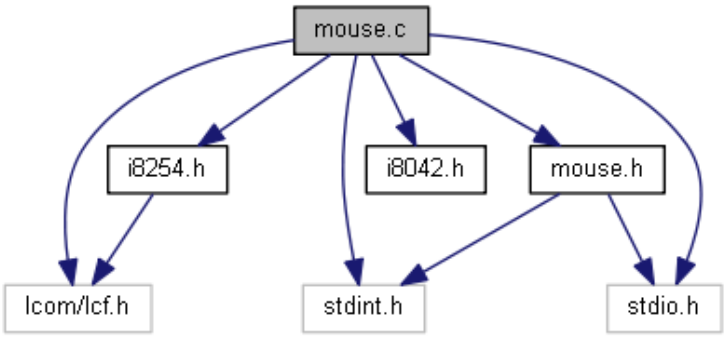
Keyboard

Neste módulo encontram-se funções para manipulação do teclado. Importado do código das aulas práticas do laboratório 3.



Mouse

Neste módulo encontram-se funções para manipulação do rato. Importado do código das aulas práticas do laboratório 4.



4. Secção IV: Detalhes da Implementação

Disclaimer: Todo o código utilizado no projeto, com a exceção do providenciado para os laboratórios de LCOM, foi desenvolvido por nós.

Como recomendado, decidimos utilizar código por camadas de modo a facilitar a utilização e escrita de funções mais complexas, assim como por questões de organização e compreensão.

Os gráficos são todos originados a partir de imagens do tipo xpm. Nestes sprites estão incluídos os botões dos menus, as diferentes fases da forca e todas as letras, números e caracteres gerados um a um.

Event-driven programming

O projeto é todo desenvolvido com base no paradigma de event-driven programming. Assim, o programa evolui de acordo com as ações do utilizador (i.e mexer o rato, clicar numa tecla…). Exceto no caso de o jogador perder por tempo.

Object-oriented programming

Neste projeto é também abordado o tema paradigma de object-oriented programming, mais especificamente no módulo list, que inclui uma struct e alguma funções associadas só e apenas a essa struct.

Lógica do Jogo

As palavras são escolhidas aleatoriamente, com base nas funções *srand()* e *rand()*, que fazem parte da biblioteca stdlib.h. A função *srand()* utiliza o número de segundos desde 1 de janeiro de 1970 para formar uma "seed", que depois é utilizada pela função *rand()* para calcular um número aleatório entre 0 e 466323 (número de palavras no ficheiro *words.txt* -1), chamar-lhe-emos *n*. Neste ponto, abre se o ficheiro *words.txt* em modo de leitura e lê-se *n* caracteres '\n', que representam o fim de linha, e seleciona-se a palavra seguinte. Verifica-se se a palavra tem a dificuldade selecionada, se tiver, o programa prossegue, senão, repete-se este processo até a palavra escolhida ter a dificuldade pretendida.

Neste ponto cria-se uma string com o tamanho da palavra a adivinhar, mas composta apenas por '~', que será impressa no ecrã e uma string com todas as letras já utilizadas, inicialmente vazia. Quando o jogador premir uma tecla pode acontecer uma das seguintes situações:

Situação 1: A tecla representa uma letra.

Situação 1.1: A letra ainda não foi usada.

Situação 1.1.1: A letra faz parte da palavra.

Resultado: Os '~' que estão no lugar da letra na palavra são substituídos pela letra e a letra é adicionada ao vetor das letras usadas;

Situação 1.1.1.1: A palavra está completa.

Resultado: Aparece uma mensagem de congratulações no ecrã;

Situação 1.1.1.1.1: A palavra é um highscore.

Resultado: Aparece uma mensagem a remarcar este acontecimento e a data e a dificuldade da palavra são guardadas no ficheiro *best.txt.*

Situação 1.1.1.2: A palavra não está complete

Resultado: Repete o processo;

Situação 1.1.2: A letra não faz parte da palavra.

Resultado: A contagem de erros aumenta em 1 e a letra é adicionada ao vetor das letras usadas;

Situação 1.2: A letra já foi usada.

Resultado: Aparece uma mensagem a avisar que a letra já foi usada;

Situação 2: A tecla não representa uma letra.

Resultado: Aparece uma mensagem a avisar que a tecla não corresponde a uma letra;

Pontuações

Para a determinação das dificuldades das palavras, ou seja, das pontuações dos jogadores foi implementado o seguinte algoritmo:

```
i=0

while(i<tamanho(palavra))

{
        pont=pont+palavra[i]
        i++

}

pont=pont*tamanho(palavra)
```

Onde *pont* representa a pontuação ou dificuldade da palavra e *tamanho(palavra)* representa o tamanho da palavra. Palavras com pontuação igual ou inferior a 50 são consideradas fáceis, com pontuação entre 51 e 175 são consideradas médias, entre 176 e 300 são consideradas difíceis e acima de 300 são consideradas extreme.

Problemas e Resoluções

Dada a dimensão do projeto, é natural que nos deparemos com alguns problemas no desenvolvimento do código, alguns dos quais foram alvo de interesse e que exigiram soluções menos ortodoxas.

Uma das resoluções que provavelmente mais chama a atenção, presente na imagem abaixo, retirada da parte final da função *proj_main_loop* no ficheiro *proj.c.* Tal resolução foi necessária para colmatar uma falha no código, dado que a partir de certo momento, o teclado ainda tinha as interrupções ativas, mas estas não estavam a ser utilizadas, pois o jogo já tinha acabado, seja por perda ou vitória. Assim, se o utilizador clicasse nas teclas durante esse período e depois saísse do jogo, as letras introduzidas ficavam guardadas no buffer e seriam "descarregadas" no próxima comando do terminal. Desta forma, além de irritante, este bug poderia causar alguns problemas de segurança se explorado por um utilizador mais experiente. A solução encontrada foi fazer polling de 300 letras, as 255 do buffer mais uma margem de segurança caso fossem introduzidas mais letras no teclado e, desta forma, dar "reset" ao buffer do teclado. Tal raciocínio é utilizado de forma semelhante para impedir que letras clicadas enquanto o utilizador navega nos menus sejam depois utilizadas como input do jogo.

Outro problema que se colocou no caminho foi na navegação entre os menus. Acontecia que quando o botão do rato estava premido, *click*=1, se o utilizador mexesse o rato, causando uma interrupção, sempre com o botão premido, o programa considerava de novo *click*=1, pois o botão estava, efetivamente premido, mas não havia intenção de clicar. Tal não acontecia, no entanto, se o rato se mantivesse estático com o botão premido, pois não causaria interrupções. Este problema foi resolvido com recurso a uma variável *clicka*, que era igual ao click anterior. Caso *clicka*=1 e *click*=1, ou seja, o rato está premido, deixa de ser considerado um novo clique do rato.

```
p=300;

 while(p>0)
 {
   p--;

   sys_inb(IN_BUF, &scancode);
         if (scancode & O_BUFFER_FULL) {

           sys_inb(I_O_BUFFER, &scancode);

       }
 }
 sys_outb(IN_BUF, READ_CMD);
 sys_outb(IN_BUF, I_O_BUFFER);



 sys_inb(I_O_BUFFER,&key);

 sys_outb(IN_BUF,I_O_BUFFER);
 sys_outb(I_O_BUFFER,  key |  BIT(0));
 free(aux);
 free(video_mem);
 vg_exit();
/*  tickdelay(100);
 printf("ACABOU!"); */
 return 0;
}
```

Figura 8 - Excerto do código da função *proj_main_loop()*

5. Secção V: Conclusões

Em relação ao desenvolvimento deste projeto, foi uma forma enriquecedora de colocar em prática os conhecimentos adquiridos ao longo das aulas.

Em relação à unidade curricular, não podemos menosprezar o apoio dos docentes que estão sempre disponíveis para esclarecer qualquer tipo de dúvidas. O desenvolver dos laboratórios nem sempre foi fácil porque todas as semanas tínhamos de interiorizar muita informação e, dada a complexidade de alguns tópicos abordados, a ajuda dos docentes da unidade curricular foi fundamental.

Entendemos a importância da unidade curricular, uma vez que ter contacto com este tipo de programação de baixo nível, nesta fase do percurso académico, torna-se desafiante.