|  |
| --- |
| FEUP – FACULDADE DE ENGENHERIA DA UNIVERSIDADE DO PORTO |
| cnix – Editor de Texto e Código |
| Laboratório de Computadores |
|  |
| **Duarte Duarte - 110509101 -** [**ei11101@fe.up.pt**](mailto:ei11101@fe.up.pt)  **Miguel Marques - 110509099 -** [**ei11099@fe.up.pt**](mailto:ei11099@fe.up.pt) |
| **31 de dezembro de 2012** |

|  |
| --- |
| Turma 1 – Grupo 5 – 2MIEIC |

# Índice

[Índice 1](#_Toc344777714)

[Estado final do projeto 2](#_Toc344777715)

[Arquitetura do programa 3](#_Toc344777716)

[Diagrama de invocação de funções 4](#_Toc344777717)

[Implementação 5](#_Toc344777718)

[Instruções 6](#_Toc344777719)

# Estado final do projeto

O cnix permite criar, abrir, editar e guardar até 11 ficheiros de texto (de cada vez).

Implementado:

* Abrir ficheiro para edição
* Guardar ficheiro editado
* Múltiplos separadores (cada tab corresponde a um ficheiro)
* Edição de texto
* Atalhos de teclado
* Botões clicáveis com interação do rato
* (Outras funcionalidades básicas necessárias para os pontos acima referidos)

Não implementado:

* Compilação de ficheiros a partir da interface gráfica
* Visualização dos erros de compilação na interface gráfica
* *Syntax highlighting* básico no texto

Nota: Os dois primeiros pontos estão parcialmente implementados no código, porém, fomos limitados pelo Minix relativamente à execução de comandos da *shell* (nomeadamente usando *system()* e *popen()*).

# Arquitetura do programa

O projeto contém módulos de baixo nível e contém estruturas de dados que os utilizam de modo a conseguirmos criar uma abstração dos controladores.

Tem um módulo gráfico que, como o próprio nome indica contém funções que interagem com a placa gráfica em modo gráfico, entre elas funções que desenham pixéis, retângulos, círculos, linhas e letras. Este módulo foi utilizado para desenhar a janela, os separadores que funcionam como as caixas de texto e todo o resto da interface gráfica.

Implementou-se também um módulo de interrupções que permite subscrever interrupções e, automaticamente, implementa o ciclo de interrupções com funções de *callback* para cada uma das interrupções subscritas.

Os módulos do *timer* e do *rtc* que interagem com o *Timer 0* e com o *Real-Time Clock,* respetivamente. *timer* é usado para atualizar o programa 60 vezes por segundo (frequência por defeito) e *rtc* é utilizado para obter data e tempo do sistema.

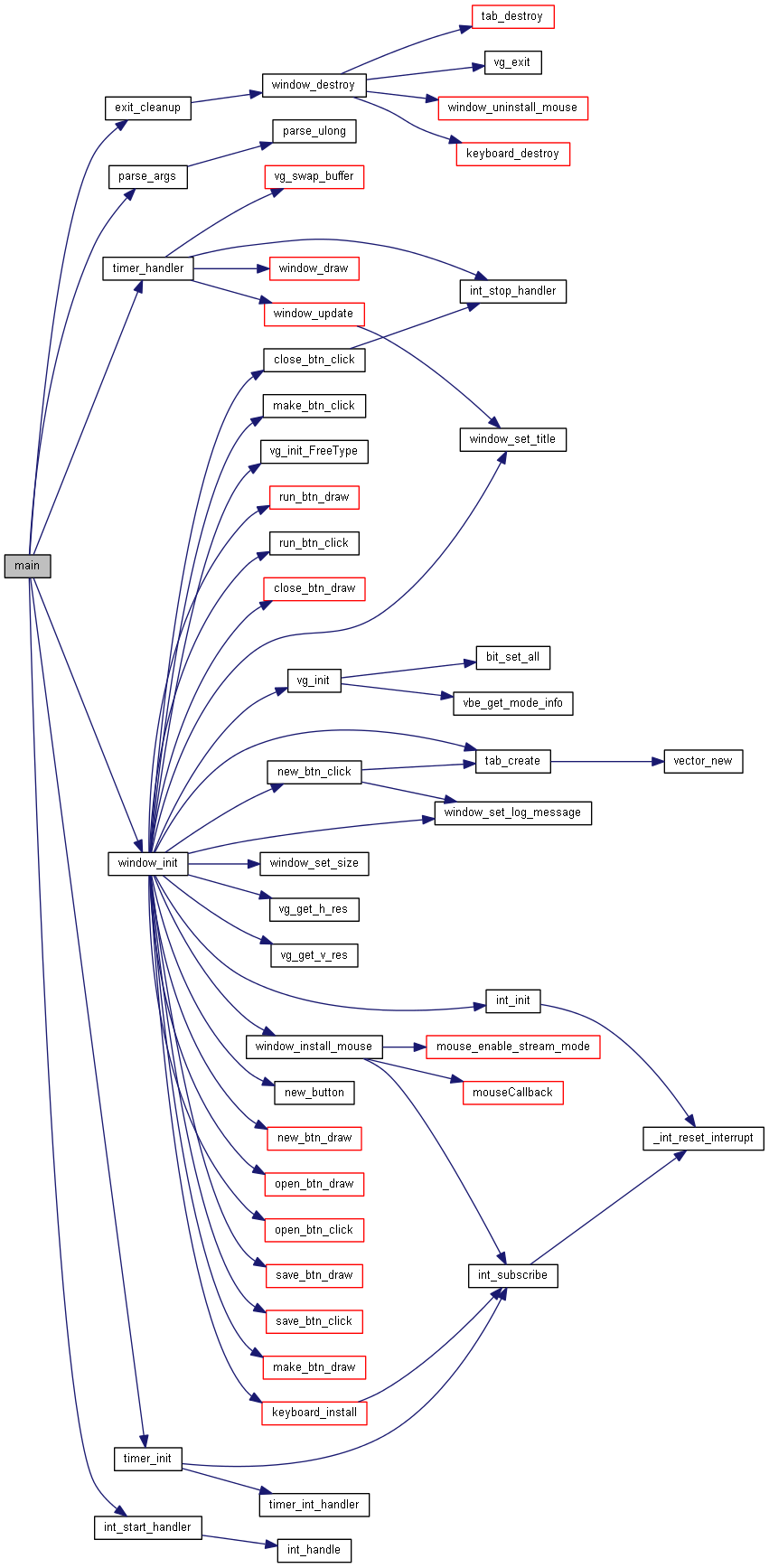
Temos ainda os módulos keyboard e mouse que utilização um módulo de baixo nível (kbc) para interagir com o teclado e com rato, respetivamente.

Dentro das estruturas de dados lógicas temos a janela (*window)*, o separador (*tab)* e o botão (*button)*. A janela é responsável pela interface gráfica e pela chamada da função de *update* dos botões e dos separadores. Os separadores são os objetos de escrita onde são apresentados os ficheiros, bem como a consola do utilizador. Os botões permitem a chamada de funções *callback* específicas.

Implementaram-se ainda contentores de dados, nomeadamente: *vector*, *list* e *priority\_list*.

(mais detalhes em */doc/html/modules.html*)

# Diagrama de invocação de funções



(Nota: gráfico gerado com Doxygen e Graphviz;

não completo devido a limitações destas ferramentas)

# Implementação

O módulo gráfico implementa *double-buffering* recorrendo a um buffer em memória virtual que é copiado para a memória física através de uma função implementada em *assembly* usando intruções *mmx*. Além disso, o módulo gráfico está preparado para funcionar em qualquer modo de cor e resolução de ecrã suportados pela VBE.

Para implementar as funcionalidades de abrir e gravar ficheiros foi implementada uma máquina de estados que permite que o programa funcione assincronamente, ou seja, o utilizador escreve o nome do ficheiro na consola do programa sem que o programa bloqueie.

Utilizámos as interrupções do *timer 0*, do teclado e do rato para interagir com esses componentes e recorremos a *polling* para obter a data e hora do rtc.

Implementámos uma funcionalidade de log, que nos permite ter uma mensagem na janela para informar o utilizador do que deve fazer e sobre erros que tenham ocorrido.

Foram implementados *arrays* dinâmicos (vector) para a implementação das tabs.

O loop principal do programa pode ser sumariado da seguinte forma:

enquanto em execução:

receber interrupção

chamar callback respetiva

timer callback:

window\_update() // define se é necessário redesenhar janela

se necessário redesenhar:

window\_draw()

swap\_buffers()

Os sucessivos *updates* e *draws* permitem que o programa não bloquei em qualquer ponto/ação, o que resulta numa interação com o utilizador agradável e funcional.

Para o desenho de caracteres para o ecrã recorreu-se à biblioteca FreeType (<http://www.freetype.org/>), o que permite o uso de diferentes de tipos de letra e de técnicas avançadas no seu desenho (por exemplo *hinting* e *kerning*).

Para o teclado e mapeamento tecla <-> caracter foi usado um *array* múltiplo indexado pelo *make scancode* da tecla, (por exemplo “*shift* + 6 = &” corresponde a *key\_c[0x36][KEY\_IDX\_SHT]*). Tal permite um acesso eficiente ao caracter pretendido (respeitando teclas modificadoras (*shift*, *alt*, *caps lock*, etc.))

# Instruções

Para abrir um ficheiro é só premir o botão representado por uma pasta amarela no canto superior direito da janela, ou escrever ‘*open’* na consola seguido de *enter*, e escrever o caminho e nome do ficheiro na consola.

Para gravar o ficheiro do separador selecionado basta carregar no botão representado por uma disquete no canto superior direito da janela, ou escrever ‘*save’* na consola seguido de *enter*, e escrever o caminho e nome do ficheiro na consola.

Para selecionar um separador pode carregar na etiqueta correspondente ou premir as teclas F1-F11. Para selecionar a consola basta carregar na mesma ou premir F12.

Para navegar no texto de um separador utilizam-se as teclas das setas ou *home*, *end*, *page up* e *page down*.