Faculdade de Engenharia da Universidade do Porto


# Minix Calendar

## *MinCAL*

Laboratório de Computadores 2017/2018

Mestrado Integrado em Engenharia Informática e Computação

Professor Pedro Alexandre Guimarães Lobo Ferreira Souto

Grupo T3G10:

Daniel Pereira da Silva up201503212@fe.up.pt

Frederico Portugal Pinho Rocha up201408030@fe.up.pt

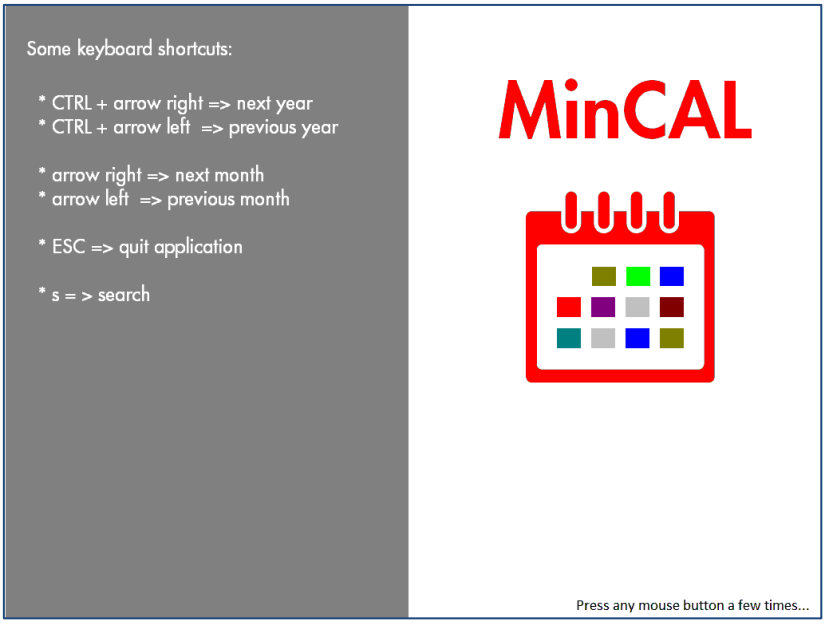
# ÍNDICE

# 1 INSTRUÇÕES DE UTILIZAÇÃO

## 1.1 ECRÃ INICIAL

Ao iniciar o programa, é apresentado um ecrã em jeito de *landing page* que serve para introduzir o utilizador ao MinCAL. Aproveitamos também para sincronizar o rato de forma simples e relativamente transparente.

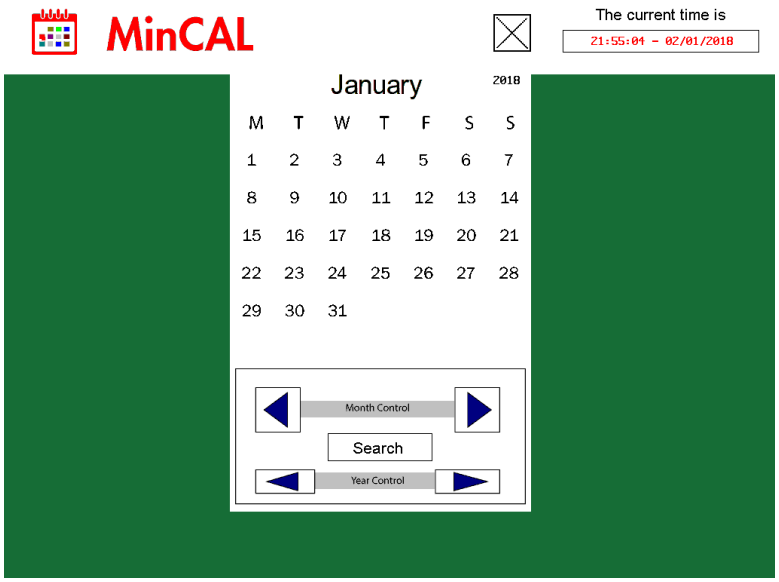
São visíveis algumas dicas de como utilizar mais facilmente a aplicação assim que se chega ao ecrã seguinte.




*1-1 Ecrã Inicial*

## 1.2 VISTA DE MÊS

Feita a sincronização (ou pressionada qualquer tecla), o utilizador é levado do ecrã inicial para a vista de calendário. No centro do ecrã é desenhado, então, o calendário. No topo do ecrã são indicadas informações adicionais, tal como o titulo do programa, a data atual e horas, e uma 'cruz' para encerrar o programa.




*1-2 Vista de Mês*

### 1.2.1 Navegar no Calendário

De modo a navegar no calendário, estão disponíveis várias opções, a saber:

- Botões presentes no ecrã (a selecionar utilizando o rato):
  - Estes são autoexplicativos.
- Setas esquerda e direita do teclado:
  - A tecla da seta esquerda muda o calendário para o mês anterior, enquanto a seta direita muda o calendário para o mês seguinte. Se premir adicionalmente a tecla CTRL, as setas do teclado ganham a funcionalidade de alterar o ano do calendário.
- Roda de deslocamento do rato:
  - A roda de deslocamento do rato pode ser utilizada conjuntamente com a tecla CTRL, de modo a alterar o ano do calendário (em vez do mês).

## 1.3 Pesquisa de Datas (via teclado)

É possível pesquisar datas utilizando o botão *Search*, ou premindo o atalho ***s***. O par mês-ano deve ser introduzido conforme indicado no ecrã. A pesquisa é iniciada clicando no botão *Ok*, ou carregando na tecla ***Enter***.

## 1.4 Terminar o Programa

De modo a terminar o programa, o utilizador pode premir a tecla ***ESC***, ou, com o rato, clicar na cruz acima do calendário.

# 2 ESTADO DO PROJETO

## 2.1 DISPOSITIVOS UTILIZADOS

| Dispositivo | Utilização | Interrupções |
|---|---|---|
| *Timer* | Ativar o refrescamento do ecrã | Sim |
| *Keyboard* | Navegar no menu. Pesquisar por meses. Com combinações de teclas. Introdução de texto (números e símbolos), com *backspace*. | Sim |
| *Mouse* | Navegar no calendário. Premir botões no ecrã. Com Deslocamento | Sim |
| *Video Card* | Desenho no ecrã. *Double-buffering*. | Não |
| *RTC* | Data e hora atual. Uso de Interrupções de *update*. | Sim |

### *2.1.1 Timer*

O *timer* é utilizado para redesenhar o ecrã a cada interrupção do Timer0.

**Vide:**

```
void timer0_int_handler();
```

### *2.1.2 Keyboard*

O teclado é utilizado para alterar os valores da estrutura do calendário. Sempre que é premida uma tecla, entra-se na condição de interrupção apresentada abaixo, onde ocorrem as verificações relacionadas as teclas premidas. O teclado é utilizado para o input de texto na caixa de pesquisa.

**Vide:**

```
KEY_PRESS* kbd_int_handler();

void start_listening(); (loop de receção de interrupções)
```

### *2.1.3 Mouse*

O rato tem vários usos no contexto do nosso projeto, e é provavelmente o periférico que mais esforço nos custou. Ora, o rato é utilizado, antes de mais, para navegar no ecrã. Deste modo, assume um papel muito semelhante ao dos ratos comuns. Adicionalmente, é possível utilizar a roda de deslocamento do rato para mudar o mês selecionado no calendário. É ainda possível, numa ação combinada do rato com o teclado, mudar de ano, e não de mês, bastando para isso premir a tecla ***CTRL*** e deslocar a referida roda.

**Vide:**

```
void check_clicks()

void start_listening(); (loop de receção de interrupções)
```

### *2.1.4 Video Card*

O modo de vídeo é utilizado para desenhar o fundo do ecrã, através da atribuição de cores aleatórias. É também utilizado para desenhar todas as *sprites* no ecrã, através de funções próprias de leitura de ficheiros *xpm*.

O modo gráfico é inicializado no modo 0x118, com uma resolução de 1024x768 e suportando cores de 24 bits. Isto faz-se recorrendo à função 0x10 do VBE (VESA BIOS Extensions). É utilizado *double-buffering* de modo a eliminar artefactos de outro modo visíveis no ecrã, dada a recorrência com que a aplicação faz o refrescamento da memória gráfica.

Usa-se deteção de colisão nos botões acionados pelo rato.

**Vide:**

```
void setP(x, y, color);

void draw_xpm_from_memory(xpm, x, y);

void draw_string(string, x, y, color);
```

### *2.1.5 Real Time Clock*

O RTC é utilizado para ler a data e as horas, atualizando-as a cada interrupção. Estas são desenhadas no ecrã utilizando uma função que foi desenvolvida para desenhar *pixmaps* de carateres individuais. As interrupções utilizadas são somente as de *update (UIE)*.

**Vide:**

```
rtc_time_t *   rtc_int_handler ();
```

# 3 ESTRUTURA DO CÓDIGO

## 3.1 MÓDULOS

### *3.1.1 proj*

Módulo que contém a função *main*. Chama a função *init*() em e_cal se o argumento passado em argv[1] for o literal *init*.

*Peso relativo: 2%*

*Principal contribuidor: Frederico Rocha*

### *3.1.2 e_cal*

Módulo responsável por inicializar o modo de video, fazer o *load* dos ficheiros xpm contidos na pasta *xpms*, chamar a função de desenho da *main page* e chamar o ciclo responsável pela resposta às interrupções.

*Peso relativo: 5%*

*Principal contribuidor: Daniel Silva*

### *3.1.3 utils*

Módulo que contém o *device listening loop.* Contém também a função que descarrega todos os XPMs para as suas devidas structs e a função que desenha a *main page*.

*Peso relativo: 20%*

*Principal contribuidor: Daniel Silva*

### *3.1.4 read_xpm*

Módulo responsável por ler ficheiros *xpm*, dado o nome do ficheiro respetivo. Define a estrutura *xpm_t*, utilizada para guardar informação relativa aos XPMs lidos de ficheiros.

*Peso relativo: 12%*

*Principal contribuidor: Daniel Silva*

### *3.1.5 mouse*

Módulo que trata de todas as ocorrências relacionadas com o rato. Contém funções feitas previamente no lab4, modificadas para acomodar este projeto. Contém também funções de deteção de colisões, tal como uma estrutura *MOUSE_ACTION* para guardar dados relativos ao rato, como posição e se certos botões estão premidos.

*Peso relativo: 10%*

*Principal contribuidor: Frederico Rocha*

### *3.1.6 keyboard*

Módulo que contém as funções de subscrição e tratamento das interrupções do teclado de aulas laboratoriais anteriores, modificadas para este projeto. Contém também a estrutura *KEY_PRESS*, responsável por guardar dados sobre os códigos da última tecla premida.

*Peso relativo: 7%*

*Principal contribuidor: Daniel Silva*

#### 3.1.7 rtc

Módulo que contém informação sobre o tempo atual, guardado numa struct *rtc_time_t.* Contém funções de subscrição e tratamento de interrupções do RTC e funções de *interrupt enabling*.

A função bcd_to_decimal() converte binário para decimal e foi retirada da internet, não tendo sido alterada de qualquer forma. URL: https://stackoverflow.com/a/42340213.

*Peso relativo: 8%*

*Principal contribuidor: Frederico Rocha*

#### *3.1.8 timer*

Este módulo contém funções de tratamento e subscrição de interrupções do timer0, adaptadas de labs anteriores.

*Peso relativo: 4%*

*Principal contribuidor: Frederico Rocha*

#### 3.1.9 vbe

Este módulo contém duas estruturas com informação sobre o *VBE Mode* e *VBE Controller*, e uma função que retorna a estrutura relativa ao modo do VBE.

*Peso relativo: 4%*

*Principal contribuidor: Daniel Silva*

#### 3.1.10 video

Módulo responsável por desenhar no ecrã. Contém funções que preenchem o *background* e que desenham *XPMs*, *strings* e caracteres específicos. Contém a estrutura *pixel_t* usada para definir cores *rgb*.

As funções de escrita de carateres na memória gráfica fazem uso de um *array*: ***letters***, que foi descarregado da internet. Link: https://stackoverflow.com/a/23130671/1469991

*Peso relativo: 14%*

*Principal contribuidor: Daniel Silva*

#### 3.1.11 video_gr

Este módulo é responsável por iniciar o modo de video. Contém funções criadas na aula laboratorial 5, reestruturadas para melhor servir este projeto, com a adição de *double buffering*.

*Peso relativo: 4%*

*Principal contribuidor: Daniel Silva*

#### 3.1.12 view

Este módulo contem informação sobre o calendário, guardada numa struct *View*, e o tratamento desta mesma. Contém também funções de desenho do calendário.

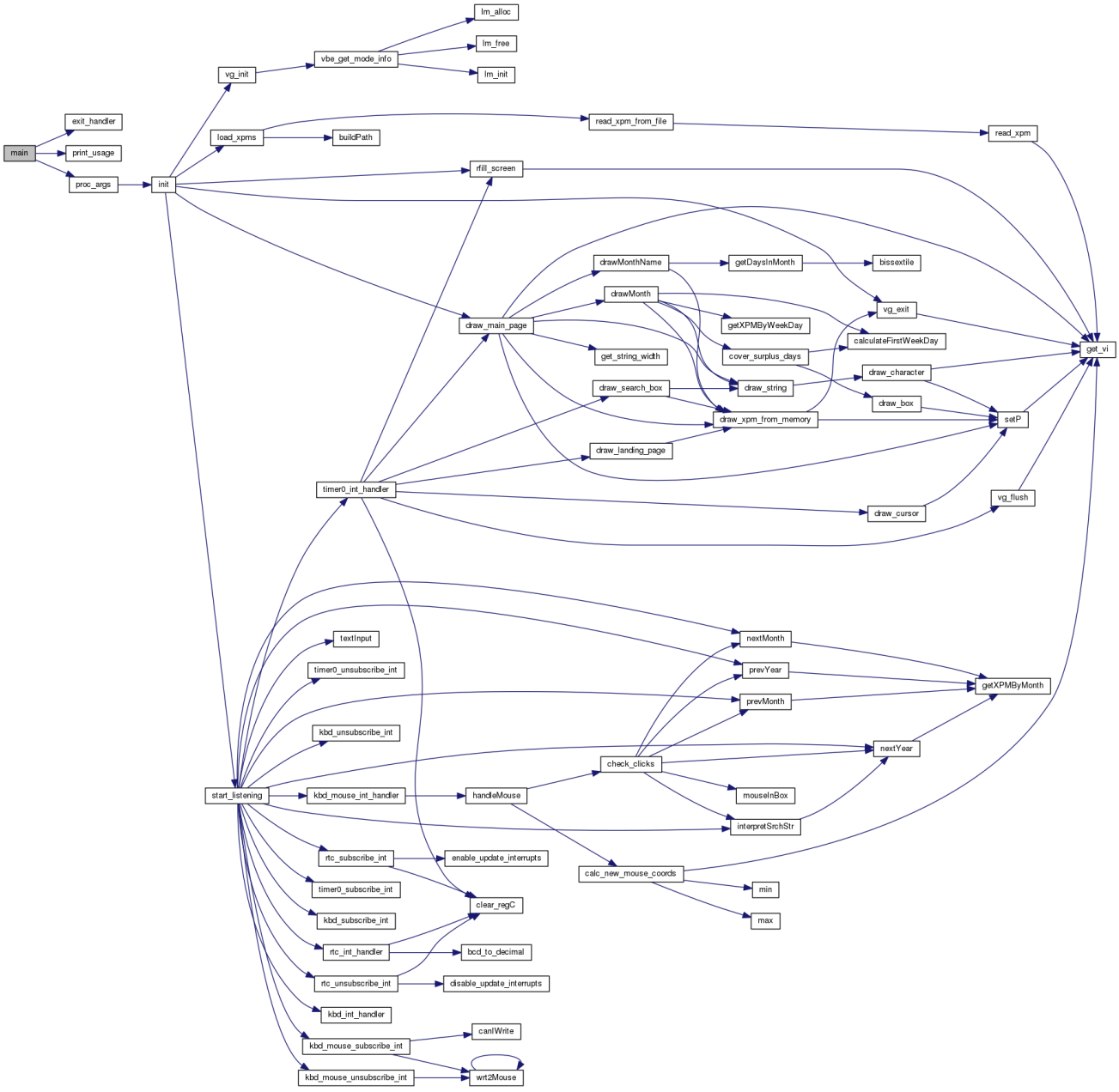
A função bissextile() indica se o ano em questão é ano bissexto e foi retirada da internet, não tendo o código sido alterado.

Link: https://github.com/FEUP-MIEEC/Prog1/blob/master/Aula_Pratica_03/Problema_8.c

*Peso relativo: 10%*

*Principal contribuidor: Frederico Rocha*

## 3.2 GRAFO DE CHAMADA DE FUNÇÕES



# 4 DETALHES DA IMPLEMENTAÇÃO

## 4.1 PONTOS DE MAIOR DIFICULDADE

### 4.1.1 Inclusão de *XPMs* na aplicação

Para a implementação das *sprites*, foi necessária alguma criatividade da nossa parte, visto que foi necessário utilizar um número de *XPMs* tal que o executável ficava tão grande que não podia ser carregado em memória para ser executado. Assim, foi necessário criar funções que lessem diretamente os ficheiros *xpm* e os guardassem em memória. Esta abordagem foi feita, e com desempenho melhor do que a anterior! Ainda assim, foi um trabalho moroso, dado que fizemos questão de utilizar algumas das funções já existentes.

## 4.2 ELEMENTOS DE VALORIZAÇÃO

### 4.2.1 Escrita de texto utilizando *XPMs* de carateres.

De modo a imprimir a data/hora do RTC, bem como o que o utilizador escreve na caixa de pesquisa, foi necessário implementar uma função capaz de escrever *strings* na memória gráfica. Para tal, recorreu-se a uma fonte de *pixmaps* obtida online (ver secção 3.1.10).

### 4.2.2 *Double-buffering* transparente

A nossa solução de *double-buffering* é extremamente transparente, de modo que as funções que escrevem para o buffer, não são informadas disto, i.e., atuam como se estivessem a escrever na memória. A função abaixo transcrita é chamada uma única vez para efetuar toda a cópia. Esta operação é pela interrupção do *timer*.

```
void vg_flush(){
#ifdef DOUBLEBUFF
	video_info_t vi = get_vi();
	memcpy(vi.rvm, vi.vm, vi.x*vi.y*vi.bpp/8);
#endif
}
```

*4-1Função responsável pela atualização da real video memory*

### 4.2.3 Utilização de combinações de teclado

Como temos feito questão de reforçar ao longo deste relatório, o premir da tecla ***CTRL*** pode alterar o funcionamento do teclado, e do rato. Consideramos este cruzamento de periféricos extremamente valioso.

## 4.3 OUTROS DETALHES

### 4.3.1 RTC

O RTC foi utilizado com base na aula laboratorial de 2013, que fizemos questão de deixar no nosso repositório (*lab6*). No nosso projeto, fizemos uso das interrupções de *update*, para fazermos a recolha da data/hora, num instante que fosse favorável à evasão de erros de leitura.

# 5 CONCLUSÕES

Inicialmente, planeávamos implementar mais uma vista para o calendário, onde se apresentava apenas os 7 dias da semana horizontalmente, com as horas no eixo vertical. Porém, com o passar do tempo apercebemo-nos que seria mais sensato não seguir essa abordagem. Parte desta ideia, fazia ainda parte a adição de um sistema de alertas, que utilizaria o alarme do RTC e o rato, para selecionar o dia ao qual se quer adicionar um lembrete. Poder-se-ia também utilizar o input do teclado para dar um título e descrição ao dito lembrete.

Uma outra funcionalidade que pensámos implementar, mas acabou por não suceder, foi um módulo de cálculo de diferença de datas.

Por outro lado, estamos felizes com o módulo de pesquisa de datas implementado.

Relativamente a críticas à UC, é o nosso entender que o uso de SVN é injustificado, num mundo onde Git é a tecnologia mais utilizada.

De outro modo, ficámos agradados em perceber que se retiraram as aulas laboratoriais do RTC e da porta série. Estas, ainda que interessantes, podem ser feitas em casa, dado que os alunos já possuem os conhecimentos necessários.

# 6 APÊNDICE

Apesar de utilizarmos ficheiros, não há qualquer cuidado a ter com a execução do programa. *Make* é suficiente para compilar, e depois o lançamento do serviço é feito como de costume. Deve ser passado o argumento “init”.