

DEPARTAMENTO DE ELECTROTECNIA

GESTOR DE INTERFACES GRÁFICAS

(Relatório de estágio PRODEP)

ISABEL MARIA CORREIA DE CASTRO

10 de Setembro de 1993



4

601.3(044-3) LEEC 1992/GASI
25 09 07

**GESTOR
DE
INTERFACES
GRÁFICAS**



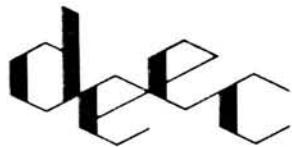
1126

Parecer

Confirmo tanto o empenho revelado no estágio pela aluna do 5º ano de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto, **Isabel Maria Correia de Castro**, como a qualidade técnica do trabalho realizado "Gestor de Interfaces Gráficas".

Porto e INESC, 15 de Setembro de 1993

Ademar Manuel Teixeira de Aguiar
Investigador do INESC



FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Departamento de Engenharia Electrotécnica e de Computadores

Rua dos Bragas, 4099 Porto Codex, PORTUGAL

Telef. 351-2-317105/107/412/457 · Telex 27323 FEUP P · Telefax 351-2-319280

Parecer

Confirmo tanto o empenho revelado no estágio pela aluna do 5º ano de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto, **Isabel Maria Correia de Castro**, como a qualidade e o interesse científico do trabalho realizado e intitulado "Gestor de Interfaces Gráficas".

Porto e FEUP, 15 de Setembro de 1993

A handwritten signature in black ink, appearing to read 'Raul Fernando Almeida Moreira Vidal'.

Raul Fernando Almeida Moreira Vidal
Professor Associado da FEUP

Agradecimentos :

- * Prof. Raul Vidal
- * Prof. ^{Dra.} Ademar Aguiar

Com a colaboração de:

- * Prof. José Manuel Mendonça
- * Eng. João José Ferreira
- * Prodep
- * Faculdade de Engenharia da
Universidade do Porto
- * Inesc

Sumário :

- Introdução
- 1^a Parte - Descrição da parte gráfica da interface
- 2^a Parte - A folha de cálculo
- 3^a Parte - Sagres
- 4^a Parte - Núcleo do programa
- 5^a Parte - Scripts
- Anexos :
 - projeto.uil
 - script
 - send_mail
 - coloca_mensagem
 - funções.c
 - projeto.c

Introdução

Este trabalho consiste na modelização de aplicações interactivas, ou seja na geração de interfaces gráficas e respectiva gestão.

Ao utilizador, é dada a possibilidade de descrever, completamente, o aspecto da sua interface gráfica através de uma linguagem simples, de programação por objectos chamada 'UIL' (User Interface Language OSF/Motif). Desta forma pode, por exemplo, ser definido o aspecto de uma janela e dos seus componentes (textos, botões, etiquetas,).

Para além de definir a interface, pretende-se que o utilizador possa associar acções a determinados eventos ocorridos, quando da utilização da mesma. Isto é, uma função será chamada quando, por exemplo, um botão for premido. Tal associação é feita num script (um ficheiro), onde é utilizada uma linguagem simples (do tipo 'IF THEN').

Com este trabalho, pretende-se desenvolver uma aplicação, que permita fazer a interacção entre o utilizador e o núcleo de uma folha de cálculo (propagador de dependências) e o sistema de gestão de regras (SAGRES). Tal aplicação terá de ter uma grande flexibilidade para permitir a sua utilização com diversas interfaces e scripts.

1^a Parte - Descrição da parte estática da interface

Numa 1^a fase, o utilizador pode definir todo o aspecto da sua interface. Para o fazer tem duas hipóteses:

- Utilizar um editor gráfico chamado 'Vuit', o qual permite que a interface seja construída rápida e simplesmente mediante a seleção e posicionamento directo, dos elementos da interface. Como saída, este programa produzirá o respectivo ficheiro em 'UIL' (User Interface Language OSF/Motif). Para esta hipótese não há necessidade de o utilizador ter conhecimento daquela linguagem.

- Construir directamente o ficheiro em 'UIL', no qual são definidos, para cada objecto, os seus argumentos (que caracterizam as suas propriedades), os objectos 'filhos' e os 'callbacks'. Como exemplo, dá-se o caso de uma janela com um texto e três botões:

```
janela1: XmBulletinBoardDialog
{
    arguments
    {
        XmNx = 750;
        XmNy = 0;
        XmNwidth = 210;
        XmNheight = 150;
        XmNnoResize = true;
        XmNresizePolicy = XmRESIZE_NONE;
        XmNautoUnmanage = false;
        XmNdefaultPosition = false;
        XmNdialogStyle = XmDIALOG_MODELESS;
    };
    control
    {
        XmText texjan1;
        XmPushButton bot1jan1;
```

```
XmPushButton bot2jan1;
XmPushButton bot3jan1;
};

callbacks
{
    MrmNcreateCallback = procedures
    {
        create_proc();
    };
};

};
```

Um tipo de descrição semelhante terá de ser feito para os objectos controlados por esta janela ('filhos'), nomeadamente para o texto texjan1 e para os botões bot1jan1, bot2jan1 e bot3jan1. O tipo de argumentos será diferente, consoante o elemento gráfico que se pretende descrever (por ex., botão, janela,...).

Posteriormente, este ficheiro será compilado para um ficheiro com extensão 'UID', o qual constituirá uma das entradas, do programa deste trabalho.

Em anexo, encontra-se a listagem do ficheiro 'projecto.uil', que foi criado como base para o desenvolvimento do programa.

```
callbacks
{
  MrmNcreateCallback = procedures
  {
    create_proc();
  };
};
```

Um tipo de descrição semelhante terá de ser feito para os objectos controlados por esta janela ('filhos'), nomeadamente para o texto texjan1 e para os botões bot1jan1, bot2jan1 e bot3jan1. O tipo de argumentos será diferente, consoante o elemento gráfico que se pretende descrever (por ex. , botão, janela,...).

Posteriormente, este ficheiro será compilado para um ficheiro com extensão 'UID', o qual constituirá uma das entradas, do programa deste trabalho.

Em anexo, encontra-se a listagem do ficheiro 'projeto.uil', que foi criado como base para o desenvolvimento do programa.

2^a Parte - A folha de cálculo

O programa funcionará ligado a um gestor de folhas de cálculo, o qual funciona como um centro propagador de dependências. Este permite a utilização de várias folhas, podendo cada uma destas ser constituída por diversas células. No programa essas folhas são usadas para representar objectos, e as suas células representam propriedades relativas ao objecto da folha a que pertencem.

A utilização das células, das folhas, permite fazer a gestão dinâmica da interface gráfica. Uma determinada função é chamada quando se verificar um certo evento, na célula em causa. Por exemplo, se o botão 1, da janela 1, for premido, implicará que a célula 'activate', desse botão, sofra uma transição de valor, e consequentemente, sejam executadas os 'callbacks' respeitantes a esse evento (por ex., abrir a janela 3).

As funções que se querem chamar, quando da ocorrência de um determinado evento são definidas em regras (o conjunto destas regras constitui o script, como veremos mais adiante).

No programa, foram, também, utilizadas 3 células de uma folha de cálculo, de carácter mais geral, chamada 'worksheet0'. As células são 'before', 'during' e 'after' e vão permitir que o utilizador possa utilizar regras que só tenham validade durante parte da existência da aplicação :

- 'Before' permite que sejam definidas as acções que se querem executar no início do programa, antes mesmo, do aparecimento da interface, definida pelo utilizador, no ecrã. Tal pode ser o caso de operações de inicialização, por exemplo. As regras definidas, com base nesta célula, não serão já consideradas fora daquele período de tempo.

- Com a célula 'during' podem ser caracterizadas regras que se querem executadas, quando a aplicação da interface, do utilizador, está a correr (e apenas enquanto está a correr).
- 'After' possibilita a inclusão de regras a serem executadas, após se ter dado a ordem de fecho do programa, mas antes que este perca o controle (por ex. regras que permitem salvar folhas).

3^a Parte - SAGRES

O programa encontra-se ligado ao SAGRES. Este é o sistema responsável pela gestão das regras, que por sua vez permitem manipular a interface gráfica escolhida, pelo utilizador.

As funções utilizadas, como 'callbacks' nas regras, têm de ser definidas de forma a obedecer às exigências do SAGRES. São usadas estruturas do tipo VLIST. No corpo principal do programa é necessário explicitar todas as funções que se pretende adicionar ao SAGRES através de 'fcaddfunc()', tendo como argumento um apontador, para a função a adicionar.

No ficheiro, 'Funções.c' (incluído em anexo), encontram-se todas as funções que foram criadas e que poderão ser utilizadas pelo utilizador para manipular a sua interface gráfica. A seguir encontra-se uma lista dessas funções, bem como uma explicação das mesmas :

- A função 'touch' simula uma transição de estado do valor da célula, de '0' para '1', a qual representará a ocorrência de um evento.

A ocorrência de um evento relativo a uma dada célula é que permitirá, depois, que o SAGRES se encarregue de mandar executar os 'callbacks' respeitantes àquela célula.

A função começa por separar o argumento, em folha de cálculo e respectiva célula. Se essa folha ainda não existir, ela é aberta. Se ainda não existir uma célula, com esse nome, nessa folha, é acrescentada essa célula à folha. Seguidamente, a célula é posta a '0' e depois a '1'.

- 'Open_janela' é a função que permite que a janela, indicada como argumento, apareça no ecrã.

É verificado se existe alguma janela com esse nome, na hierarquia de elementos gráficos da interface. Caso não exista, a função retorna 'Essa janela não existe'. Caso exista, e ainda não tenha sido aberta, são verificados quais os 'filhos' dessa janela, e esta é colocada no ecrã. Se a célula 'during', da folha de cálculo 'worksheet0', estiver a '0' ela é colocada a '1'. No caso de existir, e já ter sido aberta anteriormente, é feito o 'raisewindow' da janela, aparecendo esta em 1º plano.
Nos 2 últimos casos, a função retorna 'Janela aberta'.

- 'Close_janela' faz com que a janela, em argumento, desapareça do ecrã.

É verificado se a janela, em argumento, já foi aberta. Em caso afirmativo, retorna 'Essa janela ainda não foi aberta'. Caso contrário, a janela é retirada do ecrã e a função devolve a mensagem 'Janela fechada'.

- A função 'get_text' permite que seja lido o conteúdo, de um texto de uma janela, num determinado momento.

Esta função admite 2 parâmetros: janela e texto (para prever o caso uma janela poder ter mais que um texto). No caso de não ser indicada a janela existem 3 situações possíveis :

- Se não existir nenhuma janela aberta, a função retorna com 'Não existe nenhuma janela aberta!'.
- Se existir uma aberta, mas que não contenha nenhum 'filho' do tipo texto, retorna 'Não existe nenhuma janela na qual se possa escrever!'.
- Caso haja uma janela aberta, com um componente texto, o que estiver contido neste é lido para uma célula chamada conteúdo.

Se for dada, como argumento, o nome de uma janela, uma das seguintes hipóteses ocorrerá :

- Ou o nome indicado não corresponde a nenhuma das janelas abertas e escrito 'Essa janela não está aberta!'.
- Ou o nome não corresponde a uma janela, pelo que aparece a mensagem 'Esse objecto não é uma janela!'.
- Ou trata-se de uma janela que não tem nenhum 'filho' do tipo 'text', ao que corresponderá a frase 'Esse objecto não possui nenhum texto!'.

- Ou são indicados, quer a janela quer o texto de onde se pretende ler. Neste englobam-se os casos de o texto não ser um 'filho' daquela janela (o qual terá a mensagem 'A janela x não contém nenhum objecto de nome y !') ou o nome do texto indicado não corresponder a um elemento do tipo 'text' (ao que corresponderá a frase 'O objecto x não é do tipo texto !').

- Por fim, e não se verificando nenhum dos anteriores, o que estiver no texto será copiado para a célula conteúdo.

- 'Put_text' coloca a mensagem indicada como 1º parâmetro, na janela indicada no 2º, e no texto especificado no 3º parâmetro (no caso de a janela, em causa, ter mais que um elemento do tipo texto).

1º Caso - não é discriminada a janela - 2 situações possíveis, de falhar :

- Se nenhuma estiver aberta, escreverá 'Não existe nenhuma janela aberta !'.

- Se não houver uma janela aberta com um elemento do tipo 'text', aparecerá 'Não existe nenhuma janela aberta na qual se possa escrever !'.

2º Caso - É indicada a janela - 3 situações possíveis, de falhar :

- Não estar aberta, pelo que retornará 'Essa janela não está aberta !'.

- Não ser o nome de uma janela, ao que corresponderá 'Esse objecto não é uma janela !'.

- Não ter textos como componentes, pelo que teremos a mensagem 'Esse objecto não possui nenhum texto !'.

3º Caso - São indicados a janela e o texto (e nenhuma das opções anteriores se ter verificado) - 2 situações de falhar :

- O texto indicado não corresponder a nenhum dos componentes da janela, pelo que escreverá 'A janela x não contém nenhum objecto de nome y !'.

- O texto especificado não ser do tipo 'text', ao que corresponderá 'O objecto x não é do tipo texto !'.

Se não falhar nos casos anteriores, a janela aparecerá em 1º plano com o conteúdo que foi especificado.

- A função 'write' possibilita que seja escrito algo que o utilizador discriminou.

- 'avalia' implica que um determinado script venha a ser considerado e interpretado.

Se o ficheiro, do script indicado como argumento existir, ele é aberto para leitura e é feita a avaliação e compilação das regras nele contidas, e retornando 'Ok'.

- A função 'sair' faz o 'touch' da célula 'after' da folha de cálculo 'worksheet0' e é responsável pelo término da execução da aplicação.

- Por fim, 'sendmail' possibilita o envio de correio, para outros utilizadores.

Cada utilizador é caracterizado por 2 nomes, um corresponde à família e o outro ao membro.

Esta função tem vários parâmetros. Os 4 primeiros são a famílias e os membros do utilizador que envia o correio e do destinatário. O último parâmetro é a mensagem que se quer enviar.

4^a Parte - Núcleo do programa

O núcleo do programa, encontra-se no ficheiro 'projecto.c' (também incluído em anexo). Neste, é feito o registo das funções utilizadas como 'callbacks', pelos elementos gráficos definidos no ficheiro 'projecto.uil'.

É definida um apontador para uma estrutura com 2 campos, chamado elemento. Um dos campos é um apontador para o nome do elemento gráfico e o outro é do tipo 'widget'.

Um 'widget' é a representação de um elemento gráfico.

Os apontadores para 'elemento' é que serão a base quer de construção de uma lista com todos os 'widgets' que já foram chamados, até uma determinada altura, quer de operações de busca na mesma lista.

Após se terem feito algumas inicializações, é chamada a função 'init_application'. Esta começa por inicializar os nomes dos ficheiros utilizados: 'uid_file' é inicializado com o mesmo nome do programa, enquanto que o 'script_file' é deixado em branco.

São adicionadas (com 'scaddfunc') as funções que vão poder ser utilizadas nos scripts.

Para proceder à execução do programa, o utilizador escreve o nome do executável, seguido de alguns parâmetros opcionais: projecto [-s script] [-u uid] [-f familia] [-m membro]. O tratamento dessas opções é, também, feito nesta função.

Se for indicado um script, e o acesso a este for possível, ele é copiado para o script_file, caso contrário aparecerá a seguinte mensagem 'Não consigo abrir o ficheiro x'.

Se for indicada a opção -u, o nome que se lhe seguir é copiado para o 'uid_file'.

Se forem indicadas as opções -f e -m, elas são copiadas para as variáveis familia e membro. Se não, a familia e membro do utilizador, para efeitos de utilização de 'mail', ficam a ser 'TEST' e 'prog_test', respectivamente. Seguidamente será feito o registo, com estes identificadores, na 'mailbox'.

O ficheiro uid é procurado, e se o programa não conseguir ter acesso a ele, escreverá 'Não existe ficheiro uid.'

Se o 'script_file' não for nulo, será feita a sua interpretação através da função avalia.

Seguidamente é conhecida a hierarquia dos elementos gráficos, com base no ficheiro 'uid'. Caso ocorra algum erro, será escrita a seguinte mensagem de erro: 'Erro: a hierarquia não pode ser aberta.'

Uma vez aberta a hierarquia, o programa procederá à colocação de um 'prompt' da forma: 'fcal>'. É também feito o 'touch' da célula 'before', pelo que, se houver uma regra, no script indicado, que a contenha como condição, ela será já executada.

A partir deste momento, o utilizador poderá introduzir regras, directamente a partir do 'prompt', ler valores de células, das folhas de cálculo, dar valores a células, ou ainda chamar uma das funções, que também podem ser utilizadas, nos scripts (por ex. 'open_janela(janela1)').

O programa entrará em ciclo. Sempre que o utilizador introduz um comando, é chamada a função ler_coms. Esta procederá ao tratamento da expressão e, posteriormente colocará, de novo, aquele prompt ('fcal>'), e assim permitir a introdução de novo comando.

O utilizador pode também, em qualquer altura, receber uma mensagem, via correio. 'Receivemessage' é a função que faz o tratamento do que é enviado pelo correio, fazendo a separação em familia e membro do emissor, e do receptor e mensagem enviada. As informações sobre o correio enviado são guardadas na folha de cálculo mensagem. Quando chega 'mail', é feito o touch da célula 'rec' daquela folha. O texto recebido é guardado na célula mensagem.

No ficheiro 'projecto.c' estão, ainda, definidas várias funções. Três delas referem-se aos 'callbacks' dos objectos definidos no ficheiro 'uil':

- 'Create_proc' é a responsável pelo preenchimento da lista de 'widgets'.

A cada objecto, já utilizado, corresponde um elemento da lista, constituído por 2 items : nome e widget. No caso de se tratar de um objecto do tipo 'text', será feita uma chamada à função 'proc_text'.

- Na função 'proc_text' é procurada a folha de cálculo, cujo nome corresponde à junção do nome do 'pai' do widget com o seu próprio nome, separados por por um traço. Se essa folha não existir, ela é criada. Da mesma forma, se essa folha ainda não tiver uma célula chamada 'conteúdo', ela será, então, adicionada. Nessa célula é escrito o texto que estiver nesse 'widget'.
- 'Proc_call' determina a folha que corresponde à junção do nome do 'pai' do 'widget', com o seu próprio nome, e retorna o nome dessa folha unido à célula 'activate', por meio de um ponto (ex: 'janela1_bot3jan1.activate').

A função traduz permite determinar qual o 'widget', quando lhe é fornecido o nome daquele. Essa determinação é feita com base na lista de 'widgets', já mencionada anteriormente.

A função separa, tal como o nome indica, permite separar uma palavra em folha de cálculo e respectiva célula. Essa separação é feita com base na busca do ponto que as separa. Se o ponto não for encontrado, a célula passa a ser igual à palavra fornecida e a folha igual à folha actual.

5^a Parte - Scripts

Scripts são ficheiros constituídos por regras discriminadas pelo utilizador, e que lhe vão permitir manipular a interface gráfica, por ele escolhida.

Para ilustrar as suas possibilidades, foram incluídos, em anexo, 3 scripts (dois deles são chamados por um principal). Pretendeu-se, com aqueles, mostrar a sua utilidade e a facilidade de utilização introduzida por eles, relativamente ao caso de utilização de 'mail'.

O script 1 permitirá que o utilizador teste o 'mail', simplesmente atribuindo à célula 'send' o valor 1 (fazendo send=1, no 'prompt' fcal>). Em resposta a essa acção, o utilizador enviará para si próprio a mensagem 'testing'.

Para tornar mais simples a utilização de 'mail', e evitar que, sempre que o utilizador queira enviar uma mensagem, tenha que escrever 'sendmail' seguido de todos aqueles parâmetros, usa-se a janela envia_mail (descrita no ficheiro uil). Assim, o utilizador limita-se a preencher os campos, dessa janela, relativos às famílias e membros do emissor e destinatário, e o campo relativo à mensagem.

No script 1 estão especificadas regras para os botões 'OK' e 'Cancel'.

Quando o botão 'OK' é premido, ocasionará que o script 'send_mail' seja avaliado e, consequentemente, seja mandada executar a rotina 'sendmail' com os parâmetros lidos, dos campos das janelas.

Quando 'Cancel' é premido, a janela 'envia_mail' é fechada.

Uma outra janela, 'recebe_mail', é utilizada para mostrar as mensagens recebidas e por quem foram enviadas. Sempre que uma mensagem é recebida, é actuada a célula 'rec', da folha mensagem.

No script 1 está definido que, sempre que isto acontece, é aberta a janela 'recebe_mail' e é avaliado o script 'coloca_mensagem'. Da interpretação deste resulta que sejam escritos, naquela janela, a mensagem e a identificação do emissor (familia e membro).

Outras regras foram, ainda, definidas para as janelas 1 e 3. Quando for premido o botão 1, da janela 1, será aberta a janela 2. Como consequência de carregar no botão 2, daquela janela, teremos que é escrito, na célula 'b2', o conteúdo lido do texto da janela 1. Por sua vez, quando premido o botão 3, será colocado o conteúdo daquela célula, na janela 2. Quanto à janela 3, temos que, actuando no botão 1 implicará o fecho da janela 2, enquanto que o botão 2 terá como resposta a escrita de 'Foi premido o botão 2 da janela 3'.

Fica, assim, mostrada a grande utilidade e versatilidade dos scripts, que permitem, a um utilizador que não conheça a linguagem 'C', definir o modo de funcionamento da interface gráfica em causa.

ANEXOS

PROJECTO.UIL

```
module projeto
    names = case_sensitive

object
    receive_mail: XmBulletinBoardDialog
    {
        arguments
        {
            XmNx = 200;
            XmNy = 0;
            XmNwidth = 410;
            XmNheight = 380;
            XmNautoUnmanage = false;
            XmNdefaultPosition = false;
            XmNdialogStyle = XmDIALOG_MODELESS;
            XmNnoResize = true;
            XmNresizePolicy = XmRESIZE_NONE;
        };
        controls
        {
            XmText mensagem;
            XmText loc_fam;
            XmText loc_mem;
            XmLabel loc_fam_l;
            XmLabel loc_mem_l;
            XmPushButton botao_ok;
        };
        callbacks
        {
            MrmNcreateCallback = procedures
            {
                create_proc();
            };
        };
    };

    envia_mail: XmBulletinBoardDialog
    {
```

```
arguments
{
  XmNx = 200;
  XmNy = 450;
  XmNwidth = 410;
  XmNheight = 380;
  XmNautoUnmanage = false;
  XmNdefaultPosition = false;
  XmNdialogStyle = XmDIALOG_MODELESS;
  XmNnoResize = true;
  XmNresizePolicy = XmRESIZE_NONE;
};

controls
{
  XmText mensagem;
  XmText loc_fam2;
  XmText loc_mem2;
  XmText dest_fam2;
  XmText dest_mem2;
  XmLabel loc_fam_l;
  XmLabel loc_mem_l;
  XmLabel dest_fam_l;
  XmLabel dest_mem_l;
  XmPushButton botao_ok;
  XmPushButton botao_can;
};

callbacks
{
  MrmNcreateCallback = procedures
  {
    create_proc();
  };
};

loc_fam_l: XmLabel
{
  arguments
  {
    XmNlabelString = "Familia do emissor:";
    XmNx = 0;
```

```
    XmNy = 220;
};

};

loc_mem_1: XmLabel
{
arguments
{
    XmNlabelString = "Membro do emissor:";
    XmNx = 0;
    XmNy = 260;
};
};

dest_fam_1: XmLabel
{
arguments
{
    XmNlabelString = "Familia do receptor:";
    XmNx = 0;
    XmNy = 300;
};
};

dest_mem_1: XmLabel
{
arguments
{
    XmNlabelString = "Membro do receptor:";
    XmNx = 0;
    XmNy = 340;
};
};

janela1: XmBulletinBoardDialog
{
arguments
{
    XmNx = 750;
    XmNy = 0;
    XmNwidth = 210;
```

```
XmNheight = 150;
XmNnoResize = true;
XmNresizePolicy = XmRESIZE_NONE;
XmNautoUnmanage = false;
XmNdefaultPosition = false;
XmNdialogStyle = XmDIALOG_MODELESS;
};

controls
{
    XmText texjan1;
    XmPushButton bot1jan1;
    XmPushButton bot2jan1;
    XmPushButton bot3jan1;
};
callbacks
{
    MrmNcreateCallback = procedures
    {
        create_proc();
    };
};
};

janela2: XmBulletinBoardDialog
{
arguments
{
    XmNx = 1000;
    XmNy = 0;
    XmNwidth = 210;
    XmNheight = 150;
    XmNborderWidth = 1;
    XmNnoResize = true;
    XmNresizePolicy = XmRESIZE_NONE;
    XmNautoUnmanage = false;
    XmNdefaultPosition = false;
    XmNdialogStyle = XmDIALOG_WORK_AREA;
};
controls
{
    XmText texjan2;
```

```
XmPushButton bot1jan2;
XmPushButton bot2jan2;
XmPushButton bot3jan2;
};

callbacks
{
MrmNcreateCallback = procedures
{
    create_proc();
};
};

janela3: XmBulletinBoardDialog
{
arguments
{
XmNx = 1000;
XmNy = 200;
XmNwidth = 210;
XmNheight = 150;
XmNborderWidth = 1;
XmNnoResize = true;
XmNresizePolicy = XmRESIZE_NONE;
XmNautoUnmanage = false;
XmNdefaultPosition = false;
XmNdialogStyle = XmDIALOG_WORK_AREA;
};
controls
{
XmText texjan3;
XmPushButton bot1jan3;
XmPushButton bot2jan3;
XmPushButton bot3jan3;
};
callbacks
{
MrmNcreateCallback = procedures
{
    create_proc();
};
```

```
};

texjan1: XmText
{
arguments
{
XmNx = 0;
XmNy = 0;
XmNwidth = 192;
XmNheight = 90;
XmNcolumns = 20;
XmNrows = 3;
XmNeditMode = XmMULTI_LINE_EDIT;
XmNvalue =
" JANELA 1";
};
callbacks
{
MrmNcreateCallback = procedures
{
create_proc();
};
XmNvalueChangedCallback = procedures
{
proc_text();
};
};
};

texjan2: XmText
{
arguments
{
XmNx = 0;
XmNy = 0;
XmNwidth = 192;
XmNheight = 90;
XmNcolumns = 20;
XmNrows = 3;
XmNeditMode = XmMULTI_LINE_EDIT;
```

```
XmNvalue =
" JANELA 2";
};

callbacks
{
MrmNcreateCallback = procedures
{
create_proc();
};

XmNvalueChangedCallback = procedures
{
proc_text();
};

};

texjan3: XmText
{
arguments
{
XmNx = 0;
XmNy = 0;
XmNwidth = 192;
XmNheight = 90;
XmNcolumns = 20;
XmNrows = 3;
XmNeditMode = XmMULTI_LINE_EDIT;
XmNvalue =
" JANELA 3";
};

callbacks
{
MrmNcreateCallback = procedures
{
create_proc();
};

XmNvalueChangedCallback = procedures
{
proc_text();
};

};
```

```
};

mensagem: XmText
{
arguments
{
  XmNx = 0;
  XmNy = 0;
  XmNheight = 200;
  XmNcolumns = 51;
  XmNeditMode = XmMULTI_LINE_EDIT;
};
callbacks
{
  MrmNcreateCallback = procedures
  {
    create_proc();
  };
  XmNvalueChangedCallback = procedures
  {
    proc_text();
  };
};
};

loc_fam: XmText
{
arguments
{
  XmNx = 140;
  XmNy = 220;
  XmNcolumns = 40;
  XmNsensitive = false;
  XmNeditMode = XmSINGLE_LINE_EDIT;
};
callbacks
{
  MrmNcreateCallback = procedures
  {
    create_proc();
  };
};
```

```
XmNvalueChangedCallback = procedures
{
  proc_text();
};

loc_fam2: XmText
{
  arguments
  {
    XmNx = 140;
    XmNy = 220;
    XmNcolumns = 40;
    XmNeditMode = XmSINGLE_LINE_EDIT;
  };
  callbacks
  {
    MrmNcreateCallback = procedures
    {
      create_proc();
    };
    XmNvalueChangedCallback = procedures
    {
      proc_text();
    };
  };
};

loc_mem: XmText
{
  arguments
  {
    XmNx = 140;
    XmNy = 260;
    XmNcolumns = 40;
    XmNsensitive = false;
    XmNeditMode = XmSINGLE_LINE_EDIT;
  };
  callbacks
  {
```

```
MrmNcreateCallback = procedures
{
  create_proc();
};

XmNvalueChangedCallback = procedures
{
  proc_text();
};

};

loc_mem2: XmText
{
  arguments
  {
    XmNx = 140;
    XmNy = 260;
    XmNcolumns = 40;
    XmNeditMode = XmSINGLE_LINE_EDIT;
  };
  callbacks
  {
    MrmNcreateCallback = procedures
    {
      create_proc();
    };
    XmNvalueChangedCallback = procedures
    {
      proc_text();
    };
  };
};

dest_fam2: XmText
{
  arguments
  {
    XmNx = 140;
    XmNy = 300;
    XmNcolumns = 40;
    XmNeditMode = XmSINGLE_LINE_EDIT;
  };
};
```

```
    };
  callbacks
  {
    MrmNcreateCallback = procedures
    {
      create_proc();
    };
    XmNvalueChangedCallback = procedures
    {
      proc_text();
    };
  };
};

dest_mem2: XmText
{
  arguments
  {
    XmNx = 140;
    XmNy = 340;
    XmNcolumns = 40;
    XmNeditMode = XmSINGLE_LINE_EDIT;
  };
  callbacks
  {
    MrmNcreateCallback = procedures
    {
      create_proc();
    };
    XmNvalueChangedCallback = procedures
    {
      proc_text();
    };
  };
};

bot1jan1: XmPushButton
{
  arguments
  {
    XmNx=10;
```

```
    XmNy=100;
    XmNwidth=50;
    XmNheight= 30;
    XmNlabelString = compound_string("B1");
    };
callbacks
{
    MrmNcreateCallback = procedures
    {
        create_proc();
    };
    XmNactivateCallback = procedures
    {
        proc_call();
    };
};

bot2jan1: XmPushButton
{
arguments
{
    XmNx=80;
    XmNy=100;
    XmNwidth=50;
    XmNheight= 30;
    XmNlabelString = compound_string("B2");
    };
callbacks
{
    MrmNcreateCallback = procedures
    {
        create_proc();
    };
    XmNactivateCallback = procedures
    {
        proc_call();
    };
};
```

```
bot3jan1: XmPushButton
{
    arguments
    {
        XmNx=150;
        XmNy=100;
        XmNwidth=50;
        XmNheight= 30;
        XmNlabelString = compound_string("B3");
    };
    callbacks
    {
        MrmNcreateCallback = procedures
        {
            create_proc();
        };
        XmNactivateCallback = procedures
        {
            proc_call();
        };
    };
};
```

```
bot1jan2: XmPushButton
{
    arguments
    {
        XmNx=10;
        XmNy=100;
        XmNwidth=50;
        XmNheight= 30;
        XmNlabelString = compound_string("B1");
    };
    callbacks
    {
        MrmNcreateCallback = procedures
        {
            create_proc();
        };
        XmNactivateCallback = procedures
```

```
        {
            proc_call();
        };
    };
};

bot2jan2: XmPushButton
{
arguments
{
    XmNx=80;
    XmNy=100;
    XmNwidth=50;
    XmNheight= 30;
    XmNlabelString = compound_string("B2");
};
callbacks
{
    MrmNcreateCallback = procedures
    {
        create_proc();
    };
    XmNactivateCallback = procedures
    {
        proc_call();
    };
};
};

bot3jan2: XmPushButton
{
arguments
{
    XmNx=150;
    XmNy=100;
    XmNwidth=50;
    XmNheight= 30;
    XmNlabelString = compound_string("B3");
};
```

```
callbacks
{
  MrmNcreateCallback = procedures
  {
    create_proc();
  };
  XmNactivateCallback = procedures
  {
    proc_call();
  };
};

bot1jan3: XmPushButton
{
  arguments
  {
    XmNx=10;
    XmNy=100;
    XmNwidth=50;
    XmNheight= 30;
    XmNlabelString = compound_string("B1");
  };
  callbacks
  {
    MrmNcreateCallback = procedures
    {
      create_proc();
    };
    XmNactivateCallback = procedures
    {
      proc_call();
    };
  };
};

bot2jan3: XmPushButton
{
  arguments
  {
```

```
XmNx=80;
XmNy=100;
XmNwidth=50;
XmNheight= 30;
XmNlabelString = compound_string("B2");
};

callbacks
{
    MrmNcreateCallback = procedures
    {
        create_proc();
    };
    XmNactivateCallback = procedures
    {
        proc_call();
    };
};

};
```

```
bot3jan3: XmPushButton
{
arguments
{
    XmNx=150;
    XmNy=100;
    XmNwidth=50;
    XmNheight= 30;
    XmNlabelString = compound_string("B3");
};
callbacks
{
    MrmNcreateCallback = procedures
    {
        create_proc();
    };
    XmNactivateCallback = procedures
    {
        proc_call();
    };
};

};
```

```
};

botao_ok: XmPushButton
{
arguments
{
XmNx=345;
XmNy=75;
XmNwidth=50;
XmNheight= 30;
XmNlabelString = compound_string("OK");
};
callbacks
{
MrmNcreateCallback = procedures
{
create_proc();
};
XmNactivateCallback = procedures
{
proc_call();
};
};
};

botao_can: XmPushButton
{
arguments
{
XmNx=345;
XmNy=120;
XmNwidth=50;
XmNheight= 30;
XmNlabelString = compound_string("Cancel");
};
callbacks
{
MrmNcreateCallback = procedures
{
create_proc();
};
```

```
XmNactivateCallback = procedures
{
    proc_call();
}
};

procedure
    proc_call();
    proc_text();
    create_proc();

end module;
```

SCRIPT1

```
if send then
sendmail('TEST','prog_test','TEST','prog_test',0,300,0,'testing',8)
if envia_mail_botao_ok.activate then avalia('send_mail')
if envia_mail_botao_can.activate then close('envia_mail')
if recebe_mail_botao_ok.activate then close('recebe_mail')
if mensagem.rec then open('recebe_mail');
avalia('coloca_mensagem')

if janela1_bot1jan1.activate then open('janela2')
if janela1_bo21jan1.activate then b2=get_text('janela1')
if janela1_bot3jan1.activate then eval('put_text(b2,\'janela2\'')
if janela1_bot1jan3.activate then close('janela2')
if janela1_bot2jan3.activate then write('Foi premido o botão 2 da
janela3')
```

SEND_MAIL

```
eval('sendmail(envia_mail_dest_fam2.conteudo,envia_mail_dest_
mem2.conteudo,envia_mail_loc_fam2.conteudo,envia_mail_loc_
mem2.conteudo,0,300,0,envia_mail_mensagem.conteudo,8)')
```

COLOCA_MENSAGEM

```
eval('put_text(mensagem.mensagem,\'recebe_mail\',\'mensagem\')')
eval('put_text(mensagem.local_family,\'recebe_mail\',\'loc_fam\')')
eval('put_text(mensagem.local_member,\'recebe_mail\',\'loc_mem\')')
```

FUNÇÕES.C

```
static VLIST touch(args)
VLIST args;
{
    char str[40];
    FCELL *celula;
    char s1[20];
    char s2[20];
    VLIST ret;

    strcpy(str,avstr(args));
    separa(str,s1,s2);
    if((fc=fcgetfcal(s1))==NULL) fc=fcopen(s1,0L);
    celula=fcgetcell(fc,s2);
    if(celula==NULL)
    {
        celula=fcaddcell(fc,s2);
    }
    ret=fcreadcell(celula,0L);
    fcwritecell(celula,0L,fcpushi(0));
    fcwritecell(celula,0L,fcpushi(1));
    return(ret);
}

static FCFUNCBYNAME fctouch={"touch",touch,1,"ss"};

static VLIST get_text(args)
VLIST args;
{
    char conteudo[500];
    char janela[80];
    char texto[80];
    int k,l=0;
    Widget pai,filho;

    strcpy(janela,avstr(args));
    strcpy(texto,avstr(vlcdr(args)));



```

```
if(!strcmp(janela,""))
{
    filho=NULL;
    for(k=0;k<i;k++)
        if(XtIsComposite(widget_array[k]->w))
        {
            pai=widget_array[k]->w;
            for(l=0;l<i;l++)
            {
                if(!XmIsText(widget_array[l]->w)) continue;
                if(XtParent(widget_array[l]->w)==pai) break;
            }
            if(l==i) continue; else break;
        }
    if(l==0)
    {
        strcpy(conteudo,"Nao existe nenhuma janela aberta !");
        return(fcpushs(conteudo));
    }
    filho=widget_array[l]->w;
    if(k==i)
    {
        strcpy(conteudo,
               "Nao existe nenhuma janela aberta na qual se possa
escrever !");
        return(fcpushs(conteudo));
    }
}
else
{
    if((pai=traduz(janela))==NULL)
    {
        strcpy(conteudo,"Essa janela nao esta aberta !");
        return(fcpushs(conteudo));
    }
    if(!XtIsComposite(pai))
    {
        strcpy(conteudo,"Esse objecto nao e uma janela !");
        return(fcpushs(conteudo));
    }
    if(!strcmp(texto,""))
```

```

{
for(k=0;k<i;k++)
{
if(!XmIsText(widget_array[k]->w)) continue;
if(XtParent(widget_array[k]->w)==pai) break;
}
if(k==i)
{
strcpy(conteudo,"Esse objecto nao possui nenhum texto !");
return(fcpushs(conteudo));
}
filho=widget_array[k]->w;
}
else
if((filho=XtNameToWidget(pai,texto))==NULL)
{
sprintf(conteudo,"A janela %s nao contem nenhum objecto
de nome %s !",
janela,texto);
return(fcpushs(conteudo));
}
else
if(!XmIsText(filho))
{
sprintf(conteudo,"O objecto %s nao e do tipo texto !",texto);
return(fcpushs(conteudo));
}
strcpy(conteudo,XmTextGetString(filho));
return(fcpushs(conteudo));
}

static FCFUNCBYNAME fcget_text =
{"get_text",get_text,1,"s[s[s]]"};

static VLIST put_text(args)
VLIST args;
{
char conteudo[500];

```

```
char janela[80];
char texto[80];
int k,l=0;
Widget pai,filho;

strcpy(conteudo,avstr(vlcar(args)));
args=vlcdr(args);
strcpy(janela,avstr(args));
args=vlcdr(args);
strcpy(texto,avstr(args));
if(!strcmp(janela,""))
{
for(k=0;k<i;k++)
if(XtIsComposite(widget_array[k]->w))
{
pai=widget_array[k]->w;
for(l=0;l<i;l++)
{
if(!XmIsText(widget_array[l]->w)) continue;
if(XtParent(widget_array[l]->w)==pai) break;
}
if(l==i) continue; else break;
}
if(l==0)
{
strcpy(conteudo,"Nao existe nenhuma janela aberta !");
return(fcpushs(conteudo));
}
filho=widget_array[l]->w;
if(k==i)
{
strcpy(conteudo,
      "Nao existe nenhuma janela aberta na qual se possa
escrever !");
return(fcpushs(conteudo));
}
}
else
{
if((pai=traduz(janela))==NULL)
{
```

```

strcpy(conteudo,"Essa janela nao esta aberta !");
return(fcpushs(conteudo));
}
if(!XtIsComposite(pai))
{
strcpy(conteudo,"Esse objecto nao e uma janela !");
return(fcpushs(conteudo));
}
if(!strcmp(texto,""))
{
for(k=0;k<i;k++)
{
if(!XmIsText(widget_array[k]->w)) continue;
if(XtParent(widget_array[k]->w)==pai) break;
}
if(k==i)
{
strcpy(conteudo,"Esse objecto nao possui nenhum texto !");
return(fcpushs(conteudo));
}
filho=widget_array[k]->w;
}
else
if((filho=XtNameToWidget(pai,texto))==NULL)
{
sprintf(conteudo,"A janela %s nao contem nenhum objecto
de nome %s !",
janela,texto);
return(fcpushs(conteudo));
}
else
if(!XmIsText(filho))
{
sprintf(conteudo,"O objecto %s nao e do tipo texto !",texto);
return(fcpushs(conteudo));
}
}

XRaiseWindow(XtDisplay(XtParent(pai)),XtWindow(XtParent(
pai)));
XmTextSetString(filho,conteudo);

```

```
    return(fcpushs(""));
}

static FCFUNCBYNAME fcput_text =
{"put_text",put_text,1,"ss[s[s]]"};

static VLIST close_janela(args)
VLIST args;
{
    char janela[80];
    int id=0;
    Widget w;

    strcpy(janela,avstr(vlcar(args)));
    if((w=traduz(janela))==NULL)
    {
        strcpy(janela,"Essa janela ainda nao foi aberta !");
        return(fcpushs(janela));
    }
    XtUnmapWidget(XtParent(w));
    strcpy(janela,"Janela fechada");
    return(fcpushs(janela));
}

static FCFUNCBYNAME fcnewclose =
{"close",close_janela,1,"ss"};

static VLIST open_janela(args)
VLIST args;
{
    FCELL *celula;
    char janela[80];
    long val;
    Widget w;

    strcpy(janela,avstr(vlcar(args)));
    if((w=traduz(janela))==NULL)
    {
```

```

if (MrmFetchWidget(s_MrmHierarchy, janela, toplevel_widget,
    &w, &dummy_class) != MrmSUCCESS) return(fcpushs("Essa
janela nao existe !"));
    XtManageChild(w);
    XtRealizeWidget(w);
    XtMapWidget(XtParent(w));
    fc=fcgetfc("worksheet0");
    celula=fcgetcell(fc,"during");
    if(celula==NULL) celula=fcaddcell(fc,"during");
    args=fcreadcell(celula,0L);
    val=avint(args);
    if(val==0) fcwritecell(celula,0L,fcpushi(1));
}
else
{
    XtManageChild(w);
    XtMapWidget(XtParent(w));

    XRaiseWindow(XtDisplay(XtParent(w)),XtWindow(XtParent(w
)));
}
strcpy(janela,"Janela aberta");
return(fcpushs(janela));
}

static FCFUNCBYNAME fcnewopen =
{"open",open_janela,1,"ss"};

static VLIST n_write(args)
VLIST args;
{
    for ( ; args; args=vlcdr(args))
        puts(avstr(vlcar(args)));
    return(0);
}

static FCFUNCBYNAME fcwrite={"write",n_write,1,"vs+"};

static VLIST avalia(args)
VLIST args;

```

```

{
char st[500];
EXP exp;
VLIST ret;
FCAL *fc;
FILE *in_file;

fc=fcgetfcal("worksheet0");
strcpy(st,avstr(args));
if(!access(st,R_OK))
{
fclose(stderr);
in_file=fopen(st,"r");
while(fgets(st,500,in_file)!=NULL) fccomprule(fc,st);
fclose(in_file);
}
return(fcpushs("OK"));
}

static FCFUNCBYNAME fcavalia={"avalia",avalia,1,"ss"};

static VLIST sair(args)
VLIST args;
{
touch(fcpushs("worksheet0.after"));
XtRemoveWorkProc(loop_id);
quit_proc();
}

static FCFUNCBYNAME fcsair={"sair",sair,1,"v"};

static VLIST sendmail(args)
VLIST args;
{
static char *reply;
unsigned char
mail[MAX_LEN_MESS],mess[200],sender_fam[15],sender_mem
b[15],receiver_fam[15],receiver_memb[15],time_str[10];
short erro,ack,prvlevel,res,mode_comm,
up_id_t Local,Recv;
}

```

```
long messcode;
char message[50];

bzero(mail,MAX_LEN_MESS);
bzero(mess,200);
bzero(sender_fam,15);
bzero(sender_memb,15);
bzero(receiver_fam,15);
bzero(receiver_memb,15);
bzero(time_str,10);
up_id_t_zero(&Local);
up_id_t_zero(&Recv);

if(errno=fctestargs(args,"ssssiiisi"))
{
    sprintf( message, "%s %d", "SENDMAIL: ERROR IN
ARGUMENT NUMBER ", errno );
    return( fcpushs( "ERROR" ) );
}
else
{
    strcpy(sender_fam,(char *) avstr(vlcar(args)));
    strcpy(Local.family, sender_fam);
    args=vlcdr(args);
    strcpy(sender_memb,(char *) avstr(vlcar(args)));
    strcpy(Local.member, sender_memb);
    args=vlcdr(args);
    strcpy(receiver_fam,(char *) avstr(vlcar(args)));
    strcpy(Recv.family, receiver_fam);
    args=vlcdr(args);
    strcpy(receiver_memb,(char *) avstr(vlcar(args)));
    strcpy(Recv.member, receiver_memb);
    args=vlcdr(args);
    ack=(short) avint(vlcar(args));
    args=vlcdr(args);
    messcode=(long) avint(vlcar(args));
    args=vlcdr(args);
    prvlevel=(short) avint(vlcar(args));
    args=vlcdr(args);
    reply=avstr(vlcar(args));
    strcpy(mess,(char *) avstr(vlcar(args)));
}
```

```
args=vlcdr(args);
mode_comm=(short) avint(vlcar(args));

sprintf( mail, "%s:%s;%s:%s;%ld;0!300;1!s:\"%s\",!",
Recv.family, Recv.member,
Local.family, Local.member,
time(NULL),
mess );

res=mts_send(&Local,&Recv,mode_comm,mail,strlen(mail)+1);
if(res)
{
    mts_perror("mts_send:");
    regist_mailbox();
    return(fcpushs ("ERROR") );
}
return(fcpushs(reply));
}

static FCFUNCBYNAME fcsendmail =
{"sendmail",sendmail,1,"sssssiisi"};
```

PROJECTO.C

```
#include <stdio.h>
#include <Xm/Text.h>
#include <Mrm/MrmPublic.h>
#include <sys/ioctl.h>
#include <sys	signal.h>
#include "fcal.h"
#include "mts.h"

#define MAX_LEN_MESS 512
#define INC 1
#define SEND_MAIL 1

typedef struct elem {
    char *nome;
    Widget w;
} *elemento;

elemento *widget_array;

Display      *display;
XtApplicationContext app_context;

static Widget toplevel_widget;
static MrmHierarchy s_MrmHierarchy;
static MrmType dummy_class;
char *db_filename_vec[1];
int db_filename_num;
XtWorkProcId loop_id;

static void s_error();
static void create_proc();
static void proc_call();
static void proc_text();
static void regist_mailbox();
static void receivemessage();

static int init_application();
```

```
Widget traduz();
Boolean ler_coms();
Boolean main_loop();

static MrmRegisterArg reglist[] = {
    {"create_proc", (caddr_t) create_proc},
    {"proc_call", (caddr_t) proc_call},
    {"proc_text", (caddr_t) proc_text},
};

static int reglist_num = (sizeof reglist / sizeof reglist [0]);
int i=0,j=0,k;
char ch;
char *str=NULL;
char familia[80]="";
char membro[80]="";
FILE *in_file;

FCAL *fc, *fsetfcal();
extern FCAL *Actsheet;

#include "funcoes.c"

unsigned int main(argc,argv)
int argc;
char **argv;
{

    signal(SIGUSR1,SIG_IGN);

    Vinit();
    limit();
    fcinit();
    MrmInitialize();
    toplevel_widget = XtAppInitialize(&app_context,
    "Exemplo",NULL, 0,
        &argc, argv,NULL,NULL,0);

    XtVaSetValues(toplevel_widget,XmNwidth,1,XmNheight,1,Xm
    Nx,0,XmNy,0,XmNmappedWhenManaged,False,NULL);
    init_application(argc,argv);
```

```
    if (MrmOpenHierarchy(db_filename_num,db_filename_vec,
NULL, &s_MrmHierarchy) !=MrmSUCCESS)
        s_error("Erro: a hierarquia nao pode ser aberta.");

    MrmRegisterNames(reglist, reglist_num);
    XtRealizeWidget(toplevel_widget);

    printf("\nfcal> ");
    touch(fcpushs("worksheet0.before"));

    loop_id=XtAppAddWorkProc(app_context,(XtWorkProc)
    &main_loop,0L);
    XtAppMainLoop(app_context);
}

Boolean main_loop()
{
    receivemessage();
    return(ler_coms());
}

Boolean ler_coms()
{
    EXP exp;
    VLIST ret;
    long n_char;
    FCELL *cell;

    ioctl(0,FIONREAD,&n_char);
    if(n_char==0) return(False);
    n_char=read(0,&ch,1);
    str=(char *) realloc(str,strlen(str)+1);
    if (ch=='\n')
    {
        if(j)
        {
            str[j]=0;
            if ( exp = fcexpcomp(str) )
            {
                if ( ret = fcexpeval(exp) )

```

```
        {
            printf("%s",avstr(ret));
            fcpop(ret);
        }
        fcexpfree(exp);
    }
    printf("\nfcal> ");
    j=0;
    free(str);
}
else
    str[j++]=ch;
return(False);
}

int init_application(argc,argv)
int argc;
char **argv;
{
    char *st2;
    char uid_file[80]; /* default argv[0] */
    char script_file[80]; /* default " " */
    int c,error=0;
    extern char *optarg;
    extern int optind;
    extern int opterr;

    strcpy(uid_file,argv[0]);
    strcpy(script_file,"");

    fc = fcsetfcal(fcopen("worksheet0",0L));
    fcaddfunc(&fcnewopen);
    fcaddfunc(&fcnewclose);
    fcaddfunc(&fcput_text);
    fcaddfunc(&fcget_text);
    fcaddfunc(&fcwrite);
    fcaddfunc(&fctouch);
    fcaddfunc(&fcavalia);
    fcaddfunc(&fcstart);
```

```
fcaddfunc(&fcsendmail);

opterr=0;
while((c=getopt(argc,argv,"s:u:df:m:"))!=EOF)
    switch(c)
    {
    case 'u':
        strcpy(uid_file,optarg);
        break;
    case 's':
        if(!access(optarg,R_OK))
            strcpy(script_file,optarg);
        else printf("Nao consigo ler o ficheiro %s\n",optarg);
        break;
    case 'f':
        strcpy(familia,optarg);
        break;
    case 'm':
        strcpy(membro,optarg);
        break;
    case '?':
        printf("Uso correcto: %s [-s script] [-u uid] [-f familia] [-m
membro]\n",argv[0]);
        exit(1);
    }

fclose(stderr);

if(!strcmp(familia,""))strcpy(familia,"TEST");
if(!strcmp(membro,""))strcpy(membro,"prog_test");

regist_mailbox();

st2=(char *) malloc(sizeof(uid_file)+4);
sprintf(st2,"%s.uid",uid_file);
if(!access(st2,R_OK))
{
    db_filename_vec[0]=st2;
    db_filename_num=
(sizeof(db_filename_vec)/sizeof(db_filename_vec[0]));
} else s_error("Nao existe ficheiro uid.");
```

```

    if(strcmp(script_file,"")) avalia(fcpushs(script_file));
}

Widget traduz(s)
char *s;
{
int k,found=0;
for(k=0;k<i;k++)
{
if(!strcmp(s,widget_array[k]->nome))
{
found=1;
break;
}
}
if(!found) return(NULL);
return(widget_array[k]->w);
}

int separa(string,folha,celula)
char *string;
char *folha;
char *celula;
{
int j,k,l;

j=strlen(string);
for(k=0;k<j;k++)
  if(string[k]=='.') break;
if(k==j)
{
  for(l=0;l<j;l++)
    celula[l]=string[l];
  celula[j]='\0';
  strcpy(folha,fcalname(Actsheet));
}
else
{
  for(l=0;l<k;l++)
    folha[l]=string[l];
  folha[k]='\0';
}

```

```

        for(l=k+1;l<j+1;l++)
            celula[l-k-1]=string[l];
    }
}

static void s_error(problem_string)
    char *problem_string;
{
    printf("%s\n", problem_string);
    exit(0);
}

static void proc_call(w,tag,reason)
    Widget w;
    int *tag;
    unsigned long *reason;
{
    char s[80];
    sprintf(s,"%s_%s.activate",XtName(XtParent(w)),XtName(w));
    touch(fcpushs(s));
}

static void proc_text(w,tag,reason)
    Widget w;
    int *tag;
    unsigned long *reason;
{
    FCELL *cel;
    char s[80];

    sprintf(s,"%s_%s",XtName(XtParent(w)),XtName(w));
    if((fc=fcgetfcal(s))==NULL) fc=flopen(s,0L);
    cel=fcgetcell(fc,"conteudo");
    if(cel==NULL) cel=fcaddcell(fc,"conteudo");
    fcwritecell(cel,0L,fcpushs(XmTextGetString(w)));
}

static void create_proc(w, tag, reason)
    Widget w;
    int *tag;
    unsigned long *reason;

```

```
{  
    widget_array=(elemento *)  
realloc(widget_array,(i+1)*sizeof(widget_array));  
    widget_array[i]=(struct elem *) malloc(sizeof(struct elem));  
    widget_array[i]->w = w;  
    widget_array[i]->nome=(char *) malloc(strlen(XtName(w))+1);  
    strcpy(widget_array[i++]->nome,XtName(w));  
    if(XmIsText(w)) proc_text(w,0,0);  
}  
  
quit_proc()  
{  
    fcsetfcal(fc);  
    Vexit(0);  
    exit(1);  
}  
  
void regist_mailbox()  
{  
    up_id_t local;  
    int res;  
  
    up_id_t_zero(&local);  
  
    strcpy(local.family,familia);  
    strcpy(local.member,membro);  
  
    res=mts_register(&local,MTS_MBOX_SECURE,  
MTS_COMM_ORIGI | MTS_COMM_RECIP);  
  
    if( res )  
    {  
        mts_perror("\ncan't regist in the mailbox:");  
        exit( 0 );  
    }  
    return;  
}  
  
void receivemessage()  
{
```

```
up_id_t_ptr dest_up;
up_id_t_ptr local_up;
long res;
char *dummy;
char *dummy2;
char *msg;
char *texto;
long msg_len=MAX_LEN_MESS;
FCELL *celula;

dest_up=(up_id_t_ptr) malloc(sizeof(up_id_t));
local_up=(up_id_t_ptr) malloc(sizeof(up_id_t));
up_id_t_zero(dest_up);
up_id_t_zero(local_up);
strcpy(local_up->family,familia);
strcpy(local_up->member,membro);
strcpy(dest_up->family,"*");
strcpy(dest_up->member,"*");

while(mts_mbox_size(local_up,dest_up))
{
    msg=(char *) malloc((size_t)MAX_LEN_MESS);
    res=mts_receive(local_up,dest_up,msg,&msg_len);

    if (res == -1)
    {
        regist_mailbox();
        mts_perror("\n can't receive message :");
    }
    else
    {
        dummy=strchr(msg ':');
        *dummy=0;
        strcpy(local_up->family,msg);
        dummy++;
        dummy2=strchr(dummy,';');
        *dummy2=0;
        strcpy(local_up->member,dummy);
        dummy2++;
        dummy=strchr(dummy2,';');
    }
}
```

```

*dummy=0;
strcpy(dest_up->family,dummy2);
dummy++;
dummy2=strchr(dummy,';');
*dummy2=0;
strcpy(dest_up->member,dummy);
dummy=strstr(dummy2+1L,"!s:\\\"");
texto=dummy+4L;
texto[strlen(texto)-3]=0;
if((fc=fcgetfc("mensagem"))==NULL)
fc=fopen("mensagem",0L);
celula=fcgetcell(fc,"mensagem");
if(celula==NULL) celula=fcaddcell(fc,"mensagem");
fcwritecell(celula,0L,fcpushs(texto));
celula=fcgetcell(fc,"local_family");
if(celula==NULL) celula=fcaddcell(fc,"local_family");
fcwritecell(celula,0L,fcpushs(local_up->family));
celula=fcgetcell(fc,"local_member");
if(celula==NULL) celula=fcaddcell(fc,"local_member");
fcwritecell(celula,0L,fcpushs(local_up->member));
celula=fcgetcell(fc,"dest_family");
if(celula==NULL) celula=fcaddcell(fc,"dest_family");
fcwritecell(celula,0L,fcpushs(dest_up->family));
celula=fcgetcell(fc,"dest_member");
if(celula==NULL) celula=fcaddcell(fc,"dest_member");
fcwritecell(celula,0L,fcpushs(dest_up->member));
touch(fcpushs("mensagem.rec"));
free(msg);
}
}
free(dest_up);
free(local_up);
return;
}

```



FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

BIBLIOTECA



0000101652