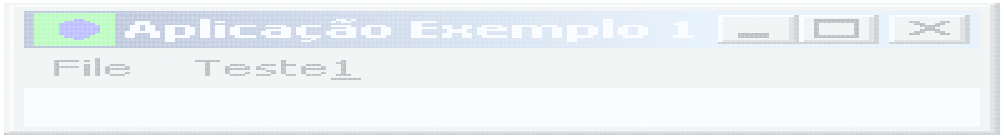


Modo GUI na API Win32

Anexo 1: Message Crackers

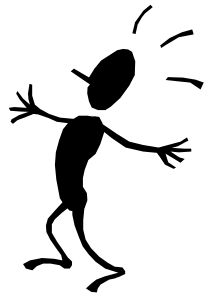
Estilo actual da WndProc

```
LRESULT CALLBACK WndProc( HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam ) {
  HDC hdc;  PAINTSTRUCT ps;  TCHAR Buffer[SIZE_BUF];
  switch (uMsg) {
    case WM_COMMAND :
      switch ( LOWORD( wParam ) ) {
        case ID_TESTE1: ++Counter; InvalidateRect(hWnd, NULL, TRUE); break;
        case ID_EXIT : DestroyWindow(hWnd); break;
        }
       break;
    case WM_PAINT :
       hdc=BeginPaint(hWnd,&ps);
        SetTextColor(hdc,RGB(0,0,0));
        _stprintf( (LPTSTR)Buffer, TEXT("Contador = %d "), Counter);
        TextOut(hdc, 20, 20, (LPCTSTR)Buffer, lstrlen((LPCTSTR)Buffer) );
       EndPaint(hWnd, &ps);
      break;
    case WM_CLOSE: break;
    case WM_DESTROY: PostQuitMessage(0); break;
    default: return (DefWindowProc(hWnd, uMsg, wParam, lParam));
  }
  return 0L;
}
```

Código complexo e propício a erros: acesso aos parâmetros, falta de *breaks*, mensagens dentro do *switch* errado, ...

Solução: ***Message Crackers***

Messages Crackers



O que são *Message Crackers*

As *Messages Crackers* são macros que se destinam a simplificar a escrita dos tratamento de eventos.

Como se utilizam as *Message Crackers*

As Messages *Crackers* são macros que são colocadas no *switch* da WinProc e que por cada mensagem chamam um procedimento com os seus parâmetros enviados como argumento

```
switch ( ) {
  case WM_COMMAND:  ... break;
  case WM_PAINT:    ... break;
  case WM_CLOSE:    ... break;
```

```
switch ( ) {
 HANDLE_MSG(hWnd, WM_COMMAND, Cls_OnCommand);
 HANDLE_MSG(hWnd, WM_PAINT,   Cls_OnPaint);
 HANDLE_MSG(hWnd, WM_CLOSE,   Cls_OnClose);
```

```
void Cls_OnCommand (HWND hWnd, int id, HWND hwndCtl, UINT codeNotify)
```

Assim o switch da WinProc é subdividido em pequenas funções, tornado o código mais legível e de mais fácil manutenção

Notas sobre as macros associadas

Macro universal que é aplicada a todas as mensagens: **HANDLE_MSG**

Macro code

```
#define HANDLE_MSG(hwnd, message, fn) \
    case (message): \
       return HANDLE_##message((hwnd), (wParam), (lParam), (fn));
```

Exemplo com: WM_COMMAND

Macro call

```
HANDLE_MSG(hwnd, WM_COMMAND, Cls_OnCommand)
```

Macro expand

```
case (WM_COMMAND):
  return HANDLE_WM_COMMAND((hwnd),(wParam),(lParam),(Cls_OnCommand));
```

Macro code

```
#define HANDLE_WM_COMMAND(hwnd, wParam, lParam, fn) \
((fn)((hwnd), (int)(LOWORD(wParam)), (HWND)(lParam), (UINT)HIWORD(wParam)), 0L)
```

Macro expand

```
case (WM_COMMAND):
 return( Cls_OnCommand(hwnd, (int)(LOWORD(wParam)), (HWND)(lParam),
                                        (UINT)HIWORD(wParam)), 0L)
```

Function called

```
void Cls_OnCommand(HWND hWnd, int id, HWND hwndCtl, UINT codeNotify)
```

Exemplo utilizando as *Messages Crackers*

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam) {
    switch (uMsg) {
      HANDLE_MSG(hWnd, WM_COMMAND, Cls_OnCommand);
      HANDLE_MSG(hWnd, WM_PAINT,   Cls_OnPaint);
      HANDLE_MSG(hWnd, WM_DESTROY, Cls_OnDestroy);
      HANDLE_MSG(hWnd, WM_CLOSE,   Cls_OnClose);
      default:
       return(DefWindowProc(hWnd, uMsg, wParam, lParam));
    } // end switch
    return 0L;
}
```

WinProc

```
void Cls_OnCommand (HWND hWnd, int id, HWND hwndCtl, UINT codeNotify) {
  switch ( id ) {
    case ID_TESTE1:
      Cont++;
      InvalidateRect(hWnd,NULL,TRUE);
      break;
    case ID_EXIT :
      DestroyWindow(hWnd);
      break;
    } // switch (id)
}
```

Cls_OnCommand

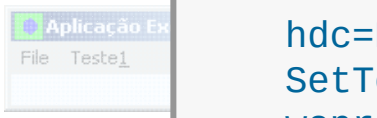
As Message Cracker estão definidas no ficheiro: WindowsX.h

Exemplo utilizando as *Messages Crackers (cont.)*

Cls_OnPaint

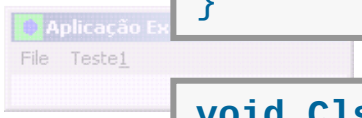
```
void Cls_OnPaint(HWND hWnd) {
    HDC hdc; PAINTSTRUCT ps;

    hdc=BeginPaint(hWnd,&ps);
    SetTextColor(hdc,RGB(0,0,0));
    wsprintf((LPTSTR)Buffer,"Contador = %d ",Cont);
    TextOut(hdc,20,20,(LPCTSTR)Buffer,lstrlen((LPCTSTR)Buffer));
    EndPaint(hWnd,&ps);
}
```



Cls_OnClose

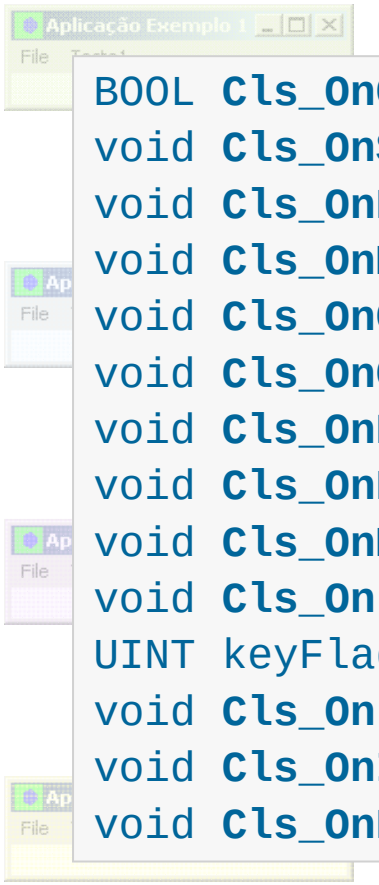
```
void Cls_OnClose(HWND hWnd) {
    MessageBox(hWnd,"Foi recebida a mensagem WM_CLOSE","DEBUG",MB_OK);
    DestroyWindow(hWnd);
}
```



Cls_OnDestroy

```
void Cls_OnDestroy(HWND hWnd) {
    MessageBox(hWnd,"Foi recebida a mensagem WM_DESTROY","DEBUG",MB_OK);
    PostQuitMessage(0);
}
```

Exemplos de *Message Crackers*



```
BOOL Cls_OnCreate(HWND hwnd, LPCREATESTRUCT lpCreateStruct);
void Cls_OnShowWindow(HWND hwnd, BOOL fShow, UINT status);
void Cls_OnPaint(HWND hwnd);
void Cls_OnMove(HWND hwnd, int x, int y);
void Cls_OnCommand(HWND hwnd, int id, HWND hwndCtl, UINT codeNotify);
void Cls_OnClose(HWND hwnd)
void Cls_OnDestroy(HWND hwnd);
void Cls_OnKey(HWND hwnd, UINT vk, BOOL fDown, int cRepeat, UINT flags);
void Cls_OnMouseMove(HWND hwnd, int x, int y, UINT keyFlags);
void Cls_OnLButtonDown(HWND hwnd, BOOL fDoubleClick, int x, int y,
UINT keyFlags);
void Cls_OnLButtonUp(HWND hwnd, int x, int y, UINT keyFlags);
void Cls_OnInitMenu(HWND hwnd, HMENU hMenu);
void Cls_OnHScroll(HWND hwnd, HWND hwndCtl, UINT code, int pos);
```

Para mais informações consultar o ficheiro windowsX.h ou Texto de apoio sobre Message Crackers

Forward de uma *Message* dentro de um *Message Cracker*

Por vezes existe a necessidade de enviar a mensagem para mais processamento, por exemplo:

- Queremos também o tratamento da `DefWinProc`, ou
- Queremos enviar a mensagem para outra *Window*

Dentro de uma *Message Cracker* teremos que fazer o *pack* dos parâmetros de volta para `wParam` e `lParam`.

Para efectuar essa tarefa e executar o *call* existem as macros:
`FORWARD_WM_* (HWND, ..., fn)`

Em que `fn` deve ser to tipo: `fn(hWnd, msg, WParam, lParam)`

```
void Cls_OnClose(HWND hWnd) {
   FORWARD_WM_CLOSE(hwnd, DefWindowProc);       // to call DefWindowProc
   // or
   HWND hOtherWindow = ...;  // get handle to other window
   FORWARD_WM_CLOSE(hOtherWindow, SendMessage)  // to send to another window
}
```