Contents

- Entrada de teclado e mouse
  - Entrada do teclado
    - Sobre a entrada do teclado
    - Usando a entrada do teclado
    - Referência de entrada do teclado
      - Funções de entrada de teclado
        - ActivateKeyboardLayout
        - BlockInput
        - Enablewindow
        - GetActiveWindow
        - Getasynckeystate
        - GetFocus
        - GetKBCodePage
        - GetKeyboardLayout
        - GetKeyboardLayoutList
        - GetKeyboardLayoutName
        - GetKeyboardState
        - GetKeyboardType
        - GetKeyNameText
        - Getkeystate
        - GetLastInputInfo
        - IsWindowEnabled
        - Keybd_event
        - LoadKeyboardLayout
        - MapVirtualKey
        - MapVirtualKeyEx
        - OemKeyScan
        - Registerhotkey
        - Sendinput

- Setactivewindow
- SetFocus
- SetKeyboardState
- ToAscii
- ToAsciiEx
- ToUnicode
- ToUnicodeEx
- UnloadKeyboardLayout
- UnregisterHotKey
- VkKeyScan
- VkKeyScanEx

Mensagens de entrada do teclado

- WM_GETHOTKEY
- Wm_sethotkey

Notificações de entrada do teclado

- Wm_activate
- WM_APPCOMMAND
- Wm_char
- Wm_deadchar
- Wm_hotkey
- Wm_keydown
- Wm_keyup
- Wm_killfocus
- Wm_setfocus
- Wm_sysdeadchar
- Wm_syskeydown
- Wm_syskeyup
- WM_UNICHAR

Estruturas de entrada de teclado

- HARDWAREINPUT
- INPUT
- KEYBDINPUT

LASTINPUTINFO
MOUSEINPUT
Constantes de entrada de teclado
códigos Virtual-Key dados
Entrada do mouse
Sobre a entrada do mouse
Usando a entrada do mouse
Referência de entrada do mouse
Funções de entrada do mouse
_TrackMouseEvent
DragDetect
GetCapture
GetDoubleClickTime
GetMouseMovePointsEx
mouse_event
Releasecapture
Setcapture
SetDoubleClickTime
SwapMouseButton
Trackmouseevent
Macros de entrada do mouse
GET_APPCOMMAND_LPARAM
GET_DEVICE_LPARAM
GET_FLAGS_LPARAM
GET_KEYSTATE_LPARAM
GET_KEYSTATE_WPARAM
GET_MOUSEORKEY_LPARAM
GET_NCHITTEST_WPARAM
GET_WHEEL_DELTA_WPARAM
GET_XBUTTON_WPARAM
Notificações de entrada do mouse
WM_CAPTURECHANGED

Wm_lbuttondblclk

WM_LBUTTONDOWN

Wm_lbuttonup

WM_MBUTTONDBLCLK

WM_MBUTTONDOWN

WM_MBUTTONUP

WM_MOUSEACTIVATE

WM_MOUSEHOVER

WM_MOUSEHWHEEL

WM_MOUSELEAVE

Wm_mousemove

WM_MOUSEWHEEL

Wm_nchittest

WM_NCLBUTTONDBLCLK

WM_NCLBUTTONDOWN

WM_NCLBUTTONUP

WM_NCMBUTTONDBLCLK

WM_NCMBUTTONDOWN

WM_NCMBUTTONUP

WM_NCMOUSEHOVER

WM_NCMOUSELEAVE

WM_NCMOUSEMOVE

WM_NCRBUTTONDBLCLK

WM_NCRBUTTONDOWN

WM_NCRBUTTONUP

WM_NCXBUTTONDBLCLK

WM_NCXBUTTONDOWN

WM_NCXBUTTONUP

WM_RBUTTONDBLCLK

Wm_rbuttondown

WM_RBUTTONUP

WM_XBUTTONDBLCLK

WM_XBUTTONDOWN
WM_XBUTTONUP
Estruturas de entrada do mouse
MOUSEMOVEPOINT
Trackmouseevent
Entrada bruta
Sobre a entrada bruta
Usando a entrada bruta
Referência de entrada bruta
Funções de entrada brutas
DefRawInputProc
GetRawInputBuffer
GetRawInputData
GetRawInputDeviceInfo
GetRawInputDeviceList
GetRegisteredRawInputDevices
RegisterRawInputDevices
Macros de Entrada Bruta
GET_RAWINPUT_CODE_WPARAM
NEXTRAWINPUTBLOCK
Notificações de entrada brutas
WM_INPUT
WM_INPUT_DEVICE_CHANGE
Estruturas de entrada brutas
RAWHID
RAWINPUT
RAWINPUTDEVICE
RAWINPUTDEVICELIST
RAWINPUTHEADER
RAWKEYBOARD
RAWMOUSE
RID_DEVICE_INFO

RID_DEVICE_INFO_HID

RID_DEVICE_INFO_KEYBOARD

RID_DEVICE_INFO_MOUSE

Entrada de teclado e mouse

15/04/2022 • 2 minutes to read

As seções a seguir descrevem métodos de captura de entrada do usuário.

Nesta seção

NOME	DESCRIÇÃO
Entrada de teclado	Discute como o sistema gera entrada de teclado e como um aplicativo recebe e processa essa entrada.
Entrada do mouse	Discute como o sistema fornece entrada de mouse para seu aplicativo e como o aplicativo recebe e processa essa entrada.
Entrada bruta	Discute como o sistema fornece entrada bruta para seu aplicativo e como um aplicativo recebe e processa essa entrada.

Entrada de teclado (entrada de teclado e mouse)

15/04/2022 • 9 minutes to read

Esta seção descreve como o sistema gera entrada de teclado e como um aplicativo recebe e processa essa entrada.

Nesta seção

NOME	DESCRIÇÃO
Sobre entrada do teclado	Discute a entrada do teclado.
Usando entrada de teclado	Aborda as tarefas associadas à entrada do teclado.
Referência de entrada do teclado	Contém a referência de API.

Funções

NOME	DESCRIÇÃO
ActivateKeyboardLayout	Define o identificador de localidade de entrada (anteriormente chamado de alça de layout do teclado) para o thread de chamada ou o processo atual. O identificador de localidade de entrada especifica uma localidade, bem como o layout físico do teclado.
BlockInput	Bloqueia eventos de entrada de teclado e mouse do alcance de aplicativos.
EnableWindow	Habilita ou desabilita a entrada do mouse e do teclado para a janela ou o controle especificado. Quando a entrada está desabilitada, a janela não recebe entradas como cliques do mouse e pressionamentos de tecla. Quando a entrada está habilitada, a janela recebe todas as entradas.
GetActiveWindow	Recupera o identificador de janela para a janela ativa anexada à fila de mensagens do thread de chamada.
GetAsyncKeyState	Determina se uma chave está ativa ou inativa no momento em que a função é chamada e se a chave foi pressionada após uma chamada anterior para **GetAsyncKeyState**.
GetFocus	Recupera o identificador para a janela que tem o foco do teclado, se a janela estiver anexada à fila de mensagens do thread de chamada.
GetKeyboardLayout	Recupera o identificador de localidade de entrada ativo (anteriormente chamado de layout de teclado) para o thread especificado. Se o parâmetro *idThread* for zero, o identificador de localidade de entrada para o thread ativo será retornado.

NOME	DESCRIÇÃO
GetKeyboardLayoutList	Recupera os identificadores de localidade de entrada (anteriormente chamados de identificadores de layout de teclado) correspondentes ao conjunto atual de localidades de entrada no sistema. A função copia os identificadores para o buffer especificado.
GetKeyboardLayoutName	Recupera o nome do identificador de localidade de entrada ativo (anteriormente chamado de layout de teclado).
GetKeyboardState	Copia o status das chaves virtuais 256 para o buffer especificado.
GetKeyNameText	Recupera uma cadeia de caracteres que representa o nome de uma chave.
GetKeyState	Recupera o status da chave virtual especificada. O status especifica se a chave está acima, abaixo ou alternada (on, off alternando cada vez que a tecla é pressionada).
GetLastInputInfo	Recupera a hora do último evento de entrada.
IsWindowEnabled	Determina se a janela especificada está habilitada para entrada de mouse e teclado.
LoadKeyboardLayout	Carrega um novo identificador de localidade de entrada (anteriormente chamado de layout de teclado) no sistema. Vários identificadores de localidade de entrada podem ser carregados de cada vez, mas apenas um por processo está ativo por vez. Carregar vários identificadores de localidade de entrada possibilita alternar rapidamente entre eles.
MapVirtualKey	Traduz (mapeia) um código de chave virtual em um código de verificação ou valor de caractere, ou traduz um código de verificação em um código de chave virtual. Para especificar um identificador para o layout de teclado a ser usado para converter o código especificado, use a função MapVirtualKeyEx .
MapVirtualKeyEx	Mapas um código de chave virtual em um código de verificação ou valor de caractere, ou traduz um código de verificação em um código de chave virtual. A função traduz os códigos usando o idioma de entrada e um identificador de localidade de entrada.
OemKeyScan	Mapas Códigos de OEMASCII 0 a 0x0FF nos códigos de verificação de OEM e Estados de deslocamento. A função fornece informações que permitem que um programa envie texto OEM para outro programa simulando a entrada do teclado.
RegisterHotKey	Define uma tecla de acesso de todo o sistema.
SendInput	Sintetiza pressionamentos de teclas, movimentos do mouse e cliques de botão.

NOME	DESCRIÇÃO
SetActiveWindow	Ativa uma janela. A janela deve ser anexada à fila de mensagens do thread de chamada.
SetFocus	Define o foco do teclado para a janela especificada. A janela deve ser anexada à fila de mensagens do thread de chamada.
Setkeyboardstate	Copia uma matriz de 256 bytes de Estados de tecla do teclado na tabela de estado de entrada do teclado do thread de chamada. Essa é a mesma tabela acessada pelas funções GetKeyboardState e GetKeyState . As alterações feitas nesta tabela não afetam a entrada do teclado para nenhum outro thread.
Toascii	Traduz o código de chave virtual especificado e o estado do teclado para o caractere ou caracteres correspondentes. A função traduz o código usando o idioma de entrada e o layout de teclado físico identificado pelo identificador de layout do teclado. Para especificar um identificador para o layout do teclado a ser usado para converter o código especificado, use a função ToAsciiEx .
ToAsciiEx	Traduz o código de chave virtual especificado e o estado do teclado para o caractere ou caracteres correspondentes. A função traduz o código usando o idioma de entrada e o layout de teclado físico identificados pelo identificador de localidade de entrada.
Tounicode	Traduz o código de chave virtual especificado e o estado do teclado para o caractere ou caracteres Unicode correspondentes. Para especificar um identificador para o layout do teclado a ser usado para converter o código especificado, use a função ToUnicodeEx .
ToUnicodeEx	Traduz o código de chave virtual especificado e o estado do teclado para o caractere ou caracteres Unicode correspondentes.
UnloadKeyboardLayout	Descarrega um identificador de localidade de entrada (anteriormente chamado de layout de teclado).
UnregisterHotKey	Libera uma tecla de acesso anteriormente registrada pelo thread de chamada.
VkKeyScanEx	Traduz um caractere para o código de chave virtual correspondente e o estado de deslocamento. A função traduz o caractere usando o idioma de entrada e o layout de teclado físico identificado pelo identificador de localidade de entrada.

As funções a seguir são obsoletas.

FUNÇÃO	DESCRIÇÃO
GetKBCodePage	Recupera a página de código atual.
_evento KEYBD	Sintetiza um pressionamento de tecla. O sistema pode usar tal pressionamento de tecla sintetizada para gerar uma mensagem do WM _ KEYUP ou WM _ KEYDOWN . O manipulador de interrupção do driver de teclado chama a função de _ evento KEYBD .
VkKeyScan	Converte um caractere para o código de chave virtual correspondente e o estado de deslocamento do teclado atual.

Mensagens

NOME	DESCRIÇÃO
WM _	Determina a tecla de acesso associada a uma janela.
WM _ SETtecla de atalho	Associa uma tecla de acesso à janela. Quando o usuário pressiona a tecla de acesso, o sistema ativa a janela.

Notificações

NOME	DESCRIÇÃO
ativar o WM _	Enviado para a janela que está sendo ativada e a janela sendo desativada. Se as janelas usarem a mesma fila de entrada, a mensagem será enviada de forma síncrona, primeiro para o procedimento de janela da janela de nível superior sendo desativada, em seguida, para o procedimento de janela da janela de nível superior que está sendo ativada. Se as janelas usarem filas de entrada diferentes, a mensagem será enviada de forma assíncrona, de modo que a janela seja ativada imediatamente.
APPCOMMAND do WM _	Notifica uma janela de que o usuário gerou um evento de comando de aplicativo, por exemplo, clicando em um botão de comando do aplicativo usando o mouse ou digitando uma tecla de comando do aplicativo no teclado.
caractere do WM _	Postado na janela com o foco do teclado quando uma mensagem do WM _ KEYDOWN é convertida pela função TranslateMessage . A mensagem do WM _ Char contém o código de caractere da chave que foi pressionada.
DEADCHAR do WM _	Postado na janela com o foco do teclado quando uma mensagem do WM _ KEYUP é convertida pela função TranslateMessage . WM _ DEADCHAR especifica um código de caractere gerado por uma chave inativa. Uma chave inativa é uma chave que gera um caractere, como o trema (ponto duplo), que é combinado com outro caractere para formar um caractere composto. Por exemplo, o caractere de trema () é gerado digitando-se a chave inativa para o caractere de trema e, em seguida, digitando a tecla o.

NOME	DESCRIÇÃO
tecla de atalho do WM _	Postado quando o usuário pressiona uma tecla de atalho registrada pela função RegisterHotKey . A mensagem é colocada na parte superior da fila de mensagens associada ao thread que registrou a tecla de acesso.
o WM _ KEYDOWN	Postado na janela com o foco do teclado quando uma tecla que não é do sistema é pressionada. Uma chave que não seja do sistema é uma chave que é pressionada quando a tecla ALT não é pressionada.
o WM _ KEYUP	Postado na janela com o foco do teclado quando uma tecla que não é do sistema é liberada. Uma chave que não seja do sistema é uma chave que é pressionada quando a tecla ALT não é pressionada ou uma tecla do teclado pressionada quando uma janela tem o foco do teclado.
KILLFOCUS do WM _	Enviado a uma janela imediatamente antes de perder o foco do teclado.
WM _ SETFOCUS	Enviado a uma janela depois de ter obtido o foco do teclado.
SYSDEADCHAR do WM _	Enviado para a janela com o foco do teclado quando uma mensagem do WM _ SYSKEYDOWN é convertida pela função TranslateMessage . WM _ SYSDEADCHAR especifica o código de caractere de uma chave inativa do sistema, ou seja, uma tecla inoperante que é pressionada enquanto mantém a tecla Alt pressionada.
SYSKEYDOWN do WM _	Postado na janela com o foco do teclado quando o usuário pressiona a tecla F10 (que ativa a barra de menus) ou mantém a tecla ALT pressionada e, em seguida, pressiona outra tecla. Ele também ocorre quando nenhuma janela tem o foco do teclado no momento; Nesse caso, a mensagem do WM _ SYSKEYDOWN é enviada para a janela ativa. A janela que recebe a mensagem pode distinguir entre esses dois contextos verificando o código de contexto no parâmetro *lParam* .
SYSKEYUP do WM _	Postado na janela com o foco do teclado quando o usuário libera uma tecla que foi pressionada enquanto a tecla ALT era mantida pressionada. Ele também ocorre quando nenhuma janela tem o foco do teclado no momento; Nesse caso, a mensagem do WM _ SYSKEYUP é enviada para a janela ativa. A janela que recebe a mensagem pode distinguir entre esses dois contextos verificando o código de contexto no parâmetro *lParam* .
UNICHAR do WM _	Postado na janela com o foco do teclado quando uma mensagem do WM _ KEYDOWN é convertida pela função TranslateMessage . A mensagem do WM _ UNICHAR contém o código de caractere da chave que foi pressionada.

Estruturas

NOME	DESCRIÇÃO

NOME	DESCRIÇÃO
HARDWAREINPUT	Contém informações sobre uma mensagem simulada gerada por um dispositivo de entrada diferente de um teclado ou mouse.
ENTRADA	Contém informações usadas para sintetizar eventos de entrada, como pressionamentos de tecla, movimento do mouse e cliques do mouse.
KEYBDINPUT	Contém informações sobre um evento de teclado simulado.
LASTINPUTINFO	Contém a hora da última entrada.
MOUSEINPUT	Contém informações sobre um evento simulado do mouse.

Constantes

NOME	DESCRIÇÃO
Códigos de chave virtual	Os nomes de constantes simbólicas, valores hexadecimais e equivalentes de mouse ou teclado para os códigos de chave virtual usados pelo sistema. Os códigos são listados em ordem numérica.

Sobre entrada do teclado

15/04/2022 • 22 minutes to read

Os aplicativos devem aceitar a entrada do usuário do teclado, bem como do mouse. Um aplicativo recebe entrada de teclado na forma de mensagens postadas em suas janelas.

Esta seção contém os seguintes tópicos:

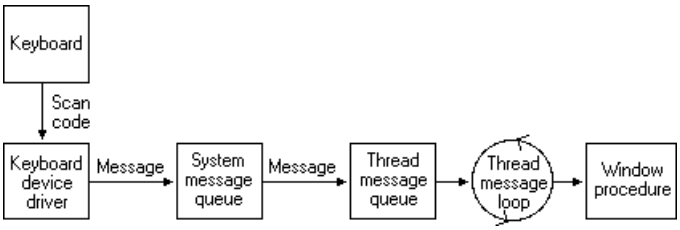
- Modelo de entrada de teclado
- Foco e ativação do teclado
- Mensagens de pressionamento de tecla
  - Pressionamentos de teclas de sistema e não sistema
  - Códigos de chave virtual descritos
  - Sinalizadores de mensagem de pressionamento de tecla
- Mensagens de caractere
  - Mensagens de caracteres não sistema
  - Mensagens de caractere morto
- Status da chave
- Pressionamento de teclas e traduções de caracteres
- Suporte a hot-key
- Teclas de teclado para navegação e outras funções
- Simulando entrada
- Idiomas, localidades e layouts de teclado

Modelo de entrada de teclado

O sistema fornece suporte de teclado independente do dispositivo para aplicativos instalando um driver de dispositivo de teclado apropriado para o teclado atual. O sistema fornece suporte a teclado independente de linguagem usando o layout de teclado específico do idioma selecionado atualmente pelo usuário ou pelo aplicativo. O driver do dispositivo de teclado recebe códigos de verificação do teclado, que são enviados para o layout do teclado em que são convertidos em mensagens e postados nas janelas apropriadas em seu aplicativo.

Atribuído a cada tecla em um teclado é um valor exclusivo chamado *código de verificação*, um identificador dependente do dispositivo para a tecla no teclado. Um teclado gera dois códigos de verificação quando o usuário digita uma tecla, uma quando o usuário pressiona a tecla e outra quando o usuário libera a tecla.

O driver do dispositivo de teclado interpreta um código de verificação e o converte (mapeia) para um *código de chave virtual*, um valor independente do dispositivo definido pelo sistema que identifica a finalidade de uma chave. Depois de traduzir um código de verificação, o layout do teclado cria uma mensagem que inclui o código de verificação, o código de chave virtual e outras informações sobre o pressionamento de teclas e, em seguida, coloca a mensagem na fila de mensagens do sistema. O sistema remove a mensagem da fila de mensagens do sistema e a posta na fila de mensagens do thread apropriado. Eventualmente, o loop de mensagem do thread remove a mensagem e a passa para o procedimento de janela apropriado para processamento. A figura a seguir ilustra o modelo de entrada do teclado.



Foco e ativação do teclado

O sistema posta mensagens de teclado na fila de mensagens do thread em primeiro plano que criou a janela com o foco do teclado. O *foco do teclado* é uma propriedade temporária de uma janela. O sistema compartilha o teclado entre todas as janelas na exibição, deslocando o foco do teclado, na direção do usuário, de uma janela para outra. A janela que tem o foco do teclado recebe (da fila de mensagens do thread que a criou) todas as mensagens de teclado até que o foco seja alterado para uma janela diferente.

Um thread pode chamar a função **GetFocus** para determinar qual de suas janelas (se houver) atualmente tem o foco do teclado. Um thread pode dar o foco do teclado a uma de suas janelas chamando a função **SetFocus** . Quando o foco do teclado muda de uma janela para outra, o sistema envia uma mensagem **WMKILLFOCUS_** para a janela que perdeu o foco e envia uma mensagem **WMSETFOCUS_** para a janela que ganhou o foco.

O conceito de foco do teclado está relacionado ao da janela ativa. A *janela ativa* é a janela de nível superior com a qual o usuário está trabalhando no momento. A janela com o foco do teclado é a janela ativa ou uma janela filho da janela ativa. Para ajudar o usuário a identificar a janela ativa, o sistema a coloca na parte superior da ordem Z e realça sua barra de título (se tiver uma) e borda.

O usuário pode ativar uma janela de nível superior clicando nela, selecionando-a usando a combinação de teclas ALT+TAB ou ALT+ESC ou selecionando-a na Lista de Tarefas. Um thread pode ativar uma janela de nível superior usando a função **SetActiveWindow** . Ele pode determinar se uma janela de nível superior que ele criou está ativa usando a função **GetActiveWindow** .

Quando uma janela é desativada e outra ativada, o sistema envia a mensagem **WMACTIVATE_**. A palavra de baixa ordem do parâmetro *wParam* será zero se a janela estiver sendo desativada e não zero se estiver sendo ativada. Quando o procedimento de janela padrão recebe a mensagem **WMACTIVATE_**, ele define o foco do teclado para a janela ativa.

Para impedir que eventos de entrada de teclado e mouse cheguem aos aplicativos, use **BlockInput**. Observe que a função **BlockInput** não interferirá na tabela de estado de entrada de teclado assíncrono. Isso significa que chamar a função **SendInput** enquanto a entrada estiver bloqueada alterará a tabela de estado de entrada do teclado assíncrono.

Mensagens de pressionamento de tecla

Pressionar uma tecla faz com que uma mensagem **WMKEYDOWN_** ou **WMSYSKEYDOWN_** seja colocada na fila de mensagens de thread anexada à janela que tem o foco do teclado. A liberação de uma chave faz com que uma mensagem **WMKEYUP_** ou **WMSYSKEYUP_** seja colocada na fila.

Mensagens de tecla para cima e teclas normalmente ocorrem em pares, mas se o usuário manter uma tecla pressionada por tempo suficiente para iniciar o recurso de repetição automática do teclado, o sistema gerará uma série de mensagens **WMKEYDOWN_** ou **WMSYSKEYDOWN_** em uma linha. Em seguida, ele gera uma única mensagem **WMKEYUP_** ou **WMSYSKEYUP_** quando o usuário libera a chave.

Esta seção contém os seguintes tópicos:

- Pressionamentos de teclas de sistema e não sistema
- Códigos de chave virtual descritos
- Sinalizadores de mensagem de pressionamento de tecla

Pressionamentos de teclas de sistema e não sistema

O sistema faz uma distinção entre pressionamentos de teclas do sistema e pressionamentos de teclas não sistema. Os pressionamentos de teclas do sistema produzem mensagens de pressionamento de tecla do sistema, **WMSYSKEYDOWN_** e **WMSYSKEYUP_**. Os pressionamentos de teclas não sistema produzem mensagens de pressionamento de tecla não sistema, **WMKEYDOWN_** e **WMKEYUP_**.

Se o procedimento de janela precisar processar uma mensagem de pressionamento de tecla do sistema, verifique se, depois de processar a mensagem, o procedimento a passará para a função **DefWindowProc** . Caso contrário, todas as operações do sistema envolvendo a tecla ALT serão desabilitadas sempre que a janela tiver o foco do teclado. Ou seja, o usuário não poderá acessar os menus da janela ou o menu sistema ou usar a combinação de teclas ALT+ESC ou ALT+TAB para ativar uma janela diferente.

As mensagens de pressionamento de tecla do sistema são principalmente usadas pelo sistema e não por um aplicativo. O sistema os usa para fornecer sua interface de teclado interna aos menus e para permitir que o usuário controle qual janela está ativa. As mensagens de pressionamento de tecla do sistema são geradas quando o usuário digita uma chave em combinação com a tecla ALT ou quando os tipos de usuário e nenhuma janela têm o foco do teclado (por exemplo, quando o aplicativo ativo é minimizado). Nesse caso, as mensagens são postadas na fila de mensagens anexada à janela ativa.

As mensagens de pressionamento de tecla não sistema são usadas pelas janelas do aplicativo; a função **DefWindowProc** não faz nada com elas. Um procedimento de janela pode descartar qualquer mensagem de pressionamento de tecla não sistema que não precise.

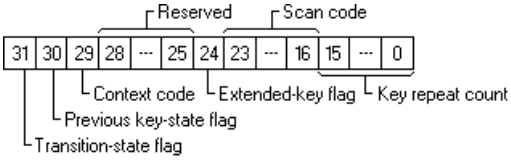
Virtual-Key códigos descritos

O parâmetro **wParam** de uma mensagem de pressionamento de tecla contém o código de chave virtual da chave que foi pressionada ou liberada. Um procedimento de janela processa ou ignora uma mensagem de pressionamento de tecla, dependendo do valor do código de chave virtual.

Um procedimento de janela típico processa apenas um pequeno subconjunto das mensagens de pressionamento de tecla que ele recebe e ignora o restante. Por exemplo, um procedimento de janela pode processar apenas mensagens de pressionamento de tecla **WMKEYDOWN_** e somente aquelas que contêm códigos de chave virtual para as chaves de movimento do cursor, as teclas de deslocamento (também chamadas de chaves de controle) e as chaves de função. Um procedimento de janela típico não processa mensagens de pressionamento de tecla a partir de chaves de caractere. Em vez disso, ele usa a função **TranslateMessage** para converter a mensagem em mensagens de caractere. Para obter mais informações sobre **TranslateMessage** e mensagens de caracteres, consulte Mensagens de Caracteres.

Sinalizadores de mensagem de pressionamento de tecla

O parâmetro **lParam** de uma mensagem de pressionamento de tecla contém informações adicionais sobre o pressionamento de tecla que gerou a mensagem. Essas informações incluem a contagem de repetição, o código de verificação, o sinalizador de chave estendida, o código de contexto, o sinalizador de estado-chave anterior e o sinalizador de estado de transição. A ilustração a seguir mostra os locais desses sinalizadores e valores no parâmetro **lParam** .



Um aplicativo pode usar os valores a seguir para manipular os sinalizadores de pressionamento de tecla.

VALOR	DESCRIÇÃO
KFALTDOWN_	Manipula o sinalizador de tecla ALT, que indica se a tecla ALT está pressionada.

VALOR	DESCRIÇÃO
KFDLGMODE_	Manipula o sinalizador do modo de diálogo, que indica se uma caixa de diálogo está ativa.
KFEXTENDED_	Manipula o sinalizador de chave estendida.
KFMENUMODE_	Manipula o sinalizador do modo de menu, que indica se um menu está ativo.
KFREPEAT_	Manipula o sinalizador de estado da chave anterior.
KFUP_	Manipula o sinalizador de estado de transição.

Código de exemplo:

```
case WM_KEYDOWN:
case WM_KEYUP:
case WM_SYSKEYDOWN:
case WM_SYSKEYUP:
{
    WORD vkCode = LOWORD(wParam);                                 // virtual-key code

    BYTE scanCode = LOBYTE(HIWORD(lParam));                       // scan code
    BOOL isExtendedKey = (HIWORD(lParam) & KF_EXTENDED) == KF_EXTENDED; // extended-key flag, 1 if scancode
has 0xe0 prefix

    BOOL repeatFlag = (HIWORD(lParam) & KF_REPEAT) == KF_REPEAT;        // previous key-state flag, 1 on
autorepeat
    WORD repeatCount = LOWORD(lParam);                            // repeat count, > 0 if several
keydown messages was combined into one message

    BOOL upFlag = (HIWORD(lParam) & KF_UP) == KF_UP;              // transition-state flag, 1 on keyup

    if (isExtendedKey)
        scanCode |= 0xe0 << 8u;

    // if we want to distinguish:
    // - VK_LSHIFT and VK_RSHIFT
    // - VK_LCONTROL and VK_RCONTROL
    // - VK_LMENU and VK_RMENU
    switch (vkCode)
    {
    case VK_SHIFT:
    case VK_CONTROL:
    case VK_MENU:
        vkCode = LOWORD(MapVirtualKeyW(scanCode, MAPVK_VSC_TO_VK_EX));
        break;
    }

    // ...
}
break;
```

Contagem repetida

Você pode verificar a contagem de repetição para determinar se uma mensagem de pressionamento de tecla representa mais de um pressionamento de tecla. O sistema incrementa a contagem quando o teclado gera mensagens WMKEYDOWN_ ou WMSYSKEYDOWN_ mais rapidamente do que um aplicativo pode processá-las. Isso geralmente ocorre quando o usuário segura uma tecla por tempo suficiente para iniciar o recurso de repetição automática do teclado. Em vez de preencher a fila de mensagens do sistema com as mensagens de

tecla suspensa resultantes, o sistema combina as mensagens em uma única mensagem de tecla para baixo e incrementa a contagem de repetição. A liberação de uma chave não pode iniciar o recurso de repetição automática, portanto, a contagem de repetição para mensagens WMKEYUP_ e WMSYSKEYUP_ é sempre definida como 1.

Digitalizar Código

O código de verificação é o valor que o hardware do teclado gera quando o usuário pressiona uma tecla. É um valor dependente do dispositivo que identifica a tecla pressionada, em oposição ao caractere representado pela chave. Um aplicativo normalmente ignora códigos de verificação. Em vez disso, ele usa os códigos de chave virtual independentes do dispositivo para interpretar mensagens de pressionamento de tecla.

Sinalizador Extended-Key

O sinalizador de tecla estendida indica se a mensagem de pressionamento de tecla se originou de uma das teclas adicionais no teclado aprimorado. As teclas estendidas consistem nas teclas ALT e CTRL no lado direito do teclado; as teclas INS, DEL, HOME, END, PAGE UP, PAGE DOWN e seta nos clusters à esquerda do teclado numérico; a chave NUM LOCK; a tecla BREAK (CTRL+PAUSE); a chave PRINT SCRN; e as teclas divide (/) e ENTER no teclado numérico. O sinalizador de chave estendida será definido se a chave for uma chave estendida.

Se especificado, o código de verificação foi precedido por um byte de prefixo com o valor 0xE0 (224).

Código de contexto

O código de contexto indica se a chave ALT estava inativa quando a mensagem de pressionamento de teclas foi gerada. O código será 1 se a chave ALT estiver inoperante e 0 se ela estiver ativada.

Sinalizador de Key-State anterior

O sinalizador de estado-chave anterior indica se a chave que gerou a mensagem de pressionamento de tecla estava anteriormente para cima ou para baixo. É 1 se a chave foi anteriormente para baixo e 0 se a chave foi anteriormente para cima. Você pode usar esse sinalizador para identificar mensagens de pressionamento de tecla geradas pelo recurso de repetição automática do teclado. Esse sinalizador é definido como 1 para mensagens de tecla WMKEYDOWN_ e WMSYSKEYDOWN_ geradas pelo recurso de repetição automática. Ele sempre é definido como 1 para mensagens WMKEYUP_ e WMSYSKEYUP_.

Sinalizador Transition-State

O sinalizador de estado de transição indica se pressionar uma chave ou liberar uma chave gerou a mensagem de pressionamento de tecla. Esse sinalizador é sempre definido como 0 para mensagens WMKEYDOWN_ e WMSYSKEYDOWN_; ele sempre é definido como 1 para mensagens WMKEYUP_ e WMSYSKEYUP_.

Mensagens de caractere

As mensagens de pressionamento de tecla fornecem muitas informações sobre pressionamentos de tecla, mas não fornecem códigos de caracteres para pressionamentos de tecla de caractere. Para recuperar códigos de caractere, um aplicativo deve incluir a função TranslateMessage em seu loop de mensagem de thread. **TranslateMessage** passa uma mensagem WMKEYDOWN_ ou WMSYSKEYDOWN_ para o layout do teclado. O layout examina o código de chave virtual da mensagem e, se corresponder a uma chave de caractere, fornece o equivalente ao código de caractere (levando em conta o estado das chaves SHIFT e CAPS LOCK). Em seguida, ele gera uma mensagem de caractere que inclui o código de caractere e coloca a mensagem na parte superior da fila de mensagens. A próxima iteração do loop de mensagem remove a mensagem de caractere da fila e envia a mensagem para o procedimento de janela apropriado.

Esta seção contém os seguintes tópicos:

- Mensagens de caractere não sistema
- Mensagens de caractere morto

Mensagens de caractere não sistema

Um procedimento de janela pode receber as seguintes mensagens de caractere: **WMCHAR_**, **WMDEADCHAR_**, **WMSYSCHAR_**, **WMSYSDEADCHAR_** e **WMUNICHAR_**. A função **TranslateMessage** gera uma mensagem **WMCHAR_** ou **WMDEADCHAR_** ao processar uma mensagem **WMKEYDOWN_**. Da mesma forma, ele gera uma mensagem **WMSYSCHAR_** ou **WMSYSDEADCHAR_** ao processar uma mensagem **WMSYSKEYDOWN_**.

Um aplicativo que processa a entrada de teclado normalmente ignora todas as mensagens **WMCHAR_** e **WMUNICHAR_**, passando qualquer outra mensagem para a função **DefWindowProc**. Observe que o **WMCHAR_** usa UTF (Formato de Transformação Unicode) de 16 bits, enquanto **o WMUNICHAR_** usa UTF-32. O sistema usa as mensagens **WMSYSCHAR_** e **WMSYSDEADCHAR_** para implementar mnemônicos de menu.

O parâmetro **wParam** de todas as mensagens de caractere contém o código de caractere da tecla de caractere que foi pressionada. O valor do código de caractere depende da classe de janela da janela que recebe a mensagem. Se a versão Unicode da função **RegisterClass** foi usada para registrar a classe de janela, o sistema fornece caracteres Unicode para todas as janelas dessa classe. Caso contrário, o sistema fornece códigos de caractere ASCII. Para obter mais informações, consulte Unicode e Conjuntos de Caracteres.

O conteúdo do parâmetro **lParam** de uma mensagem de caractere é idêntico ao conteúdo do parâmetro **lParam** da mensagem key-down que foi convertida para produzir a mensagem de caractere. Para obter informações, consulte Sinalizadores de Mensagem de Tecla.

mensagens Dead-Character

Alguns teclados não ingleses contêm teclas de caractere que não devem produzir caracteres sozinhos. Em vez disso, eles são usados para adicionar um diacrítico ao caractere produzido pelo pressionamento de tecla subsequente. Essas chaves são chamadas *de chaves mortas*. A tecla circunflexa em um teclado alemão é um exemplo de chave morta. Para inserir o caractere que consiste em um "o" com um circunflexo, um usuário alemão digitaria a chave circunflexa seguida pela chave "o". A janela com o foco do teclado receberia a seguinte sequência de mensagens:

1. **WMKEYDOWN_**
2. **WMDEADCHAR_**
3. **WMKEYUP_**
4. **WMKEYDOWN_**
5. **WMCHAR_**
6. **WMKEYUP_**

TranslateMessage gera a mensagem **WMDEADCHAR_** quando processa a mensagem **WMKEYDOWN_** de uma chave morta. Embora o parâmetro *wParam* da mensagem **WMDEADCHAR_** contenha o código de caractere do diacrítico da chave morta, um aplicativo normalmente ignora a mensagem. Em vez disso, ele processa a mensagem **WMCHAR_** gerada pelo pressionamento de tecla subsequente. O parâmetro *wParam* da mensagem **WMCHAR_** contém o código de caractere da letra com o diacrítico. Se o pressionamento de tecla subsequente gerar um caractere que não pode ser combinado com um diacrítico, o sistema gerará duas mensagens **WMCHAR_**. O parâmetro *wParam* do primeiro contém o código de caractere do diacrítico; o parâmetro *wParam* do segundo contém o código de caractere da chave de caractere subsequente.

A função **TranslateMessage** gera a mensagem **WMSYSDEADCHAR_** quando processa a mensagem **WMSYSKEYDOWN_** de uma chave morta do sistema (uma tecla morta que é pressionada em combinação com a tecla ALT). Um aplicativo normalmente ignora a mensagem **WMSYSDEADCHAR_**.

Status da chave

Ao processar uma mensagem de teclado, talvez seja necessário determinar o status de outra chave além daquela que gerou a mensagem atual. Por exemplo, um aplicativo de processamento de palavras que permite

que o usuário pressione SHIFT+END para selecionar um bloco de texto deve verificar o status da tecla SHIFT sempre que receber uma mensagem de pressionamento de tecla da tecla END. O aplicativo pode usar a função **GetKeyState** para determinar o status de uma chave virtual no momento em que a mensagem atual foi gerada; ele pode usar a função **GetAsyncKeyState** para recuperar o status atual de uma chave virtual.

O layout do teclado mantém uma lista de nomes. O nome de uma chave que produz um único caractere é o mesmo que o caractere produzido pela chave. O nome de uma chave não caracter, como TAB e ENTER, é armazenado como uma cadeia de caracteres. Um aplicativo pode recuperar o nome de qualquer chave do driver do dispositivo chamando a função **GetKeyNameText** .

Pressionamento de teclas e traduções de caractere

O sistema inclui várias funções de finalidade especiais que convertem códigos de verificação, códigos de caractere e códigos de chave virtual fornecidos por várias mensagens de pressionamento de tecla. Essas funções incluem **MapVirtualKey**, **ToAscii**, **ToUnicode** e **VkKeyScan**.

Além disso, o Microsoft Rich Edit 3.0 dá suporte ao IME HexToUnicode, que permite que um usuário converta entre caracteres hexadecimal e Unicode usando teclas quentes. Isso significa que, quando o Microsoft Rich Edit 3.0 for incorporado a um aplicativo, o aplicativo herdará os recursos do IME HexToUnicode.

Suporte Hot-Key

Uma *chave frequente* é uma combinação de chaves que gera uma mensagem **WMHOTKEY_**, uma mensagem que o sistema coloca na parte superior da fila de mensagens de um thread, ignorando todas as mensagens existentes na fila. Os aplicativos usam teclas de acesso para obter entrada de teclado de alta prioridade do usuário. Por exemplo, ao definir uma chave de acesso que consiste na combinação de teclas CTRL+C, um aplicativo pode permitir que o usuário cancele uma operação demorada.

Para definir uma chave ativa, um aplicativo chama a função **RegisterHotKey**, especificando a combinação de chaves que gera a mensagem **WMHOTKEY_**, o identificador para a janela para receber a mensagem e o identificador da chave de acesso. Quando o usuário pressiona a tecla de acesso, uma mensagem **WMHOTKEY_** é colocada na fila de mensagens do thread que criou a janela. O parâmetro *wParam* da mensagem contém o identificador da chave de acesso. O aplicativo pode definir várias teclas de acesso para um thread, mas cada chave de acesso no thread deve ter um identificador exclusivo. Antes que o aplicativo seja encerrado, ele deve usar a função **UnregisterHotKey** para destruir a chave de acesso.

Os aplicativos podem usar um controle de teclas frequentes para facilitar a escolha de uma chave frequente pelo usuário. Os controles de teclas frequentes normalmente são usados para definir uma chave quente que ativa uma janela; eles não usam as funções **RegisterHotKey** e **UnregisterHotKey** . Em vez disso, um aplicativo que usa um controle de teclas frequentes normalmente envia a mensagem **WMSETHOTKEY_** para definir a tecla de acesso. Sempre que o usuário pressiona a tecla de acesso, o sistema envia uma mensagem **WMSYSCOMMAND_** especificando SCHOTKEY_. Para obter mais informações sobre controles de teclas frequentes, consulte "Usando controles de teclas frequentes" em controles de teclas frequentes.

Teclas de teclado para navegação e outras funções

Windows fornece suporte para teclados com chaves especiais para funções de navegador, funções de mídia, inicialização de aplicativos e gerenciamento de energia. O **WMAPPCOMMAND_ dá** suporte às teclas de teclado extras. Além disso, a função **ShellProc** é modificada para dar suporte às teclas de teclado extras.

É improvável que uma janela filho em um aplicativo de componente seja capaz de implementar comandos diretamente para essas teclas de teclado extras. Portanto, quando uma dessas chaves é pressionada, **DefWindowProc** enviará uma mensagem **WMAPPCOMMAND_** para uma janela. **DefWindowProc** também colocará a mensagem **WMAPPCOMMAND_** em sua janela pai. Isso é semelhante à maneira como os menus de contexto são invocados com o botão direito do mouse, que é que **DefWindowProc** envia uma mensagem

WMCONTEXTMENU_ em um clique no botão direito e o bolhas para seu pai. Além disso, se **DefWindowProc** receber uma mensagem **WMAPPCOMMAND_** para uma janela de nível superior, ela chamará um gancho de shell com código **HSHELLAPPCOMMAND_**.

Windows também dá suporte ao Microsoft IntelliMouse Explorer, que é um mouse com cinco botões. Os dois botões extras dão suporte à navegação do navegador para frente e para trás. Para obter mais informações, consulte XBUTTONs.

Simulando entrada

Para simular uma série ininterrupta de eventos de entrada do usuário, use a função **SendInput** . A função aceita três parâmetros. O primeiro parâmetro, *cInputs*, indica o número de eventos de entrada que serão simulados. O segundo parâmetro, *rgInputs*, é uma matriz de estruturas **INPUT** , cada uma descrevendo um tipo de evento de entrada e informações adicionais sobre esse evento. O último parâmetro, *cbSize*, aceita o tamanho da estrutura **INPUT** , em bytes.

A função **SendInput** funciona injetando uma série de eventos de entrada simulados no fluxo de entrada de um dispositivo. O efeito é semelhante a chamar a função **keybdevent_** ou **mouseevent_** repetidamente, exceto que o sistema garante que nenhum outro evento de entrada intercale com os eventos simulados. Quando a chamada for concluída, o valor retornado indicará o número de eventos de entrada executados com êxito. Se esse valor for zero, a entrada será bloqueada.

A função **SendInput** não redefine o estado atual do teclado. Portanto, se o usuário tiver alguma tecla pressionada ao chamar essa função, ele poderá interferir nos eventos gerados por essa função. Se você estiver preocupado com possíveis interferências, verifique o estado do teclado com a função **GetAsyncKeyState** e corrija conforme necessário.

Idiomas, localidades e layouts de teclado

Um *idioma* é um idioma natural, como inglês, francês e japonês. Uma *sublanguagem* é uma variante de uma linguagem natural falada em uma região geográfica específica, como as sublanguages em inglês faladas no Reino Unido e a Estados Unidos. Os aplicativos usam valores, chamados de identificadores de idioma, para identificar exclusivamente idiomas e sublanguages.

Normalmente, os aplicativos usam *localidades* para definir o idioma no qual a entrada e a saída são processadas. Definir a localidade do teclado, por exemplo, afeta os valores de caractere gerados pelo teclado. Definir a localidade para a exibição ou impressora afeta os glifos exibidos ou impressos. Os aplicativos definem a localidade de um teclado carregando e usando layouts de teclado. Eles definem a localidade para uma exibição ou impressora selecionando uma fonte que dá suporte à localidade especificada.

Um layout de teclado não só especifica a posição física das teclas no teclado, mas também determina os valores de caractere gerados pressionando essas teclas. Cada layout identifica o idioma de entrada atual e determina quais valores de caractere são gerados por quais chaves e combinações de chaves.

Cada layout de teclado tem um identificador correspondente que identifica o layout e o idioma. A palavra baixa do identificador é um identificador de idioma. A palavra alta é um identificador de dispositivo, especificando o layout físico ou é zero, indicando um layout físico padrão. O usuário pode associar qualquer idioma de entrada a um layout físico. Por exemplo, um usuário de língua inglesa que trabalha muito ocasionalmente em francês pode definir o idioma de entrada do teclado como francês sem alterar o layout físico do teclado. Isso significa que o usuário pode inserir texto em francês usando o layout em inglês familiar.

Geralmente, não é esperado que os aplicativos manipulem idiomas de entrada diretamente. Em vez disso, o usuário configura combinações de idioma e layout e alterna entre elas. Quando o usuário clica no texto marcado com um idioma diferente, o aplicativo chama a função **ActivateKeyboardLayout** para ativar o layout padrão do usuário para esse idioma. Se o usuário editar texto em um idioma que não esteja na lista ativa, o aplicativo

poderá chamar a função **LoadKeyboardLayout** com o idioma para obter um layout com base nesse idioma.

A função **ActivateKeyboardLayout** define o idioma de entrada para a tarefa atual. O parâmetro *hkl* pode ser o identificador para o layout do teclado ou um identificador de idioma estendido zero. Os identificadores de layout de teclado podem ser obtidos da função **LoadKeyboardLayout** ou **GetKeyboardLayoutList** . Os valores **HKLNEXT_** e **HKLPREV_** também podem ser usados para selecionar o teclado seguinte ou anterior.

A função **GetKeyboardLayoutName** recupera o nome do layout de teclado ativo para o thread de chamada. Se um aplicativo criar o layout ativo usando a função **LoadKeyboardLayout** , **GetKeyboardLayoutName** recuperará a mesma cadeia de caracteres usada para criar o layout. Caso contrário, a cadeia de caracteres é o identificador de idioma primário correspondente à localidade do layout ativo. Isso significa que a função pode não necessariamente diferenciar entre layouts diferentes com o mesmo idioma primário, portanto, não é possível retornar informações específicas sobre o idioma de entrada. No entanto, a função **GetKeyboardLayout** pode ser usada para determinar o idioma de entrada.

A função **LoadKeyboardLayout** carrega um layout de teclado e disponibiliza o layout para o usuário. Os aplicativos podem tornar o layout imediatamente ativo para o thread atual usando o valor **KLFACTIVATE_**. Um aplicativo pode usar o valor **KLFREORDER_** para reordenar os layouts sem especificar também o valor **KLFACTIVATE_**. Os aplicativos sempre devem usar o valor **KLFSUBSTITUTEOK__** ao carregar layouts de teclado para garantir que a preferência do usuário, se houver, esteja selecionada.

Para suporte multilíngue, a função **LoadKeyboardLayout** fornece os **sinalizadores KLFREPLACELANG_** e **KLFNOTELLSHELL_**. O **sinalizador KLFREPLACELANG_** direciona a função para substituir um layout de teclado existente sem alterar o idioma. Tentar substituir um layout existente usando o mesmo identificador de idioma, mas sem especificar **KLFREPLACELANG_** é um erro. O **sinalizador KLFNOTELLSHELL_** impede que a função notifique o shell quando um layout de teclado é adicionado ou substituído. Isso é útil para aplicativos que adicionam vários layouts em uma série consecutiva de chamadas. Esse sinalizador deve ser usado em todas, exceto na última chamada.

A função **UnloadKeyboardLayout** é restrita porque não pode descarregar o idioma de entrada padrão do sistema. Isso garante que o usuário sempre tenha um layout disponível para inserir texto usando o mesmo conjunto de caracteres usado pelo shell e pelo sistema de arquivos.

Usando a entrada do teclado

15/04/2022 • 11 minutes to read

Uma janela recebe a entrada do teclado na forma de mensagens de teclas e mensagens de caractere. O loop de mensagem anexado à janela deve incluir código para converter mensagens de teclas nas mensagens de caractere correspondentes. Se a janela exibir a entrada do teclado em sua área de cliente, ela deverá criar e exibir um a tecla caret para indicar a posição em que o próximo caractere será inserido. As seções a seguir descrevem o código envolvido no recebimento, processamento e exibição da entrada do teclado:

- Ing Keystroke Messages
- Traduzindo mensagens de caractere
- Ing Character Messages
- Usando o caret
- Exibindo a entrada do teclado

Ing Keystroke Messages

O procedimento de janela da janela que tem o foco do teclado recebe mensagens de teclas quando o usuário digita no teclado. As mensagens de teclas são **WM _ KEYDOWN, WM _ KEYUP, WM _ SYSKEYDOWN** e **WM _ SYSKEYUP.** Um procedimento de janela típico ignora todas as mensagens de teclas, exceto **WM _ KEYDOWN.** O sistema posta a **mensagem WM _ KEYDOWN** quando o usuário pressiona uma tecla.

Quando o procedimento de janela recebe a mensagem **WM _ KEYDOWN,** ele deve examinar o código de chave virtual que acompanha a mensagem para determinar como processar o teclas. O código de chave virtual está no parâmetro *wParam da* mensagem. Normalmente, um aplicativo processa apenas os toques de tecla gerados por chaves não anácteres, incluindo as chaves de função, as chaves de movimento do cursor e as chaves de finalidade especial, como INS, DEL, HOME e END.

O exemplo a seguir mostra a estrutura de procedimento de janela que um aplicativo típico usa para receber e processar mensagens de teclas.

```
        case WM_KEYDOWN:
            switch (wParam)
            {
                case VK_LEFT:

                    // Process the LEFT ARROW key.

                    break;

                case VK_RIGHT:

                    // Process the RIGHT ARROW key.

                    break;

                case VK_UP:

                    // Process the UP ARROW key.

                    break;

                case VK_DOWN:

                    // Process the DOWN ARROW key.

                    break;

                case VK_HOME:

                    // Process the HOME key.

                    break;

                case VK_END:

                    // Process the END key.

                    break;

                case VK_INSERT:

                    // Process the INS key.

                    break;

                case VK_DELETE:

                    // Process the DEL key.

                    break;

                case VK_F2:

                    // Process the F2 key.

                    break;


                // Process other non-character keystrokes.

                default:
                    break;
            }
```

Traduzindo mensagens de caractere

Qualquer thread que receba a entrada de caractere do usuário deve incluir a **função TranslateMessage** em seu loop de mensagem. Essa função examina o código de chave virtual de uma mensagem de teclas e, se o código corresponde a um caractere, coloca uma mensagem de caractere na fila de mensagens. A mensagem de caractere é removida e expedida na próxima iteração do loop de mensagem; O *parâmetro wParam* da mensagem contém o código de caractere.

Em geral, o loop de mensagem de um thread deve usar a **função TranslateMessage** para converter todas as mensagens, não apenas mensagens de chave virtual. Embora **TranslateMessage** não tenha efeito sobre outros tipos de mensagens, ele garante que a entrada do teclado seja traduzida corretamente. O exemplo a seguir mostra como incluir a **função TranslateMessage** em um loop de mensagem de thread típico.

```
MSG msg;
BOOL bRet;

while (( bRet = GetMessage(&msg, (HWND) NULL, 0, 0)) != 0)
{
    if (bRet == -1);
    {
        // handle the error and possibly exit
    }
    else
    {
        if (TranslateAccelerator(hwndMain, haccl, &msg) == 0)
        {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
    }
}
```

Ing Character Messages

Um procedimento de janela recebe uma mensagem de caractere quando a **função TranslateMessage** converte um código de chave virtual correspondente a uma chave de caractere. As mensagens de caractere **são WM _ CHAR, WM _ DEADCHAR, WM _ SYSCHAR**e **WM _ SYSDEADCHAR.** Um procedimento de janela típico ignora todas as mensagens de caractere, exceto **WM _ CHAR.** A **função TranslateMessage** gera uma mensagem **WM _ CHAR** quando o usuário pressiona qualquer uma das seguintes chaves:

- Qualquer chave de caractere
- BACKSPACE
- ENTER (retorno de carro)
- ESC
- SHIFT+ENTER (linefeed)
- TAB

Quando um procedimento de janela recebe a mensagem **WM _ CHAR,** ele deve examinar o código de caractere que acompanha a mensagem para determinar como processar o caractere. O código de caractere está no parâmetro *wParam da* mensagem.

O exemplo a seguir mostra a estrutura de procedimento de janela que um aplicativo típico usa para receber e processar mensagens de caractere.

```
        case WM_CHAR:
            switch (wParam)
            {
                case 0x08:

                    // Process a backspace.

                    break;

                case 0x0A:

                    // Process a linefeed.

                    break;

                case 0x1B:

                    // Process an escape.

                    break;

                case 0x09:

                    // Process a tab.

                    break;

                case 0x0D:

                    // Process a carriage return.

                    break;

                default:

                    // Process displayable characters.

                    break;
            }
```

Usando o caret

Uma janela que recebe a entrada do teclado normalmente exibe os caracteres que o usuário digita na área de cliente da janela. Uma janela deve usar um sinal de adoção para indicar a posição na área do cliente em que o próximo caractere será exibido. A janela também deve criar e exibir o a tecla quando receber o foco do teclado e ocultar e destruir o a tecla quando perder o foco. Uma janela pode executar essas operações no processamento das mensagens **WM _ SETFOCUS** e **WM _ KILLFOCUS.** Para obter mais informações sobre os pontos de cuidado, consulte Carets.

Exibindo a entrada do teclado

O exemplo nesta seção mostra como um aplicativo pode receber caracteres do teclado, exibi-los na área do cliente de uma janela e atualizar a posição do a careta com cada caractere digitado. Ele também demonstra como mover o aro em resposta às teclas SETA PARA A ESQUERDA, SETA PARA A DIREITA, HOME e FIM e mostra como realçar o texto selecionado em resposta à combinação de teclas SHIFT+SETA PARA A DIREITA.

Durante o processamento da **mensagem WM _ CREATE,** o procedimento de janela mostrado no exemplo aloca um buffer de 64K para armazenar a entrada do teclado. Ele também recupera as métricas da fonte carregada no momento, salvando a altura e a largura média dos caracteres na fonte. A altura e a largura são usadas no processamento da mensagem **WM _ SIZE** para calcular o tamanho da linha e o número máximo de linhas, com base no tamanho da área do cliente.

O procedimento de janela cria e exibe o acento ao processar a **mensagem WM _ SETFOCUS.** Ele oculta e exclui o acento ao processar a mensagem **WM _ KILLFOCUS.**

Ao processar a mensagem **WM _ CHAR,** o procedimento de janela exibe caracteres, armazena-os no buffer de entrada e atualiza a posição do aro. O procedimento de janela também converte caracteres de tabulação em quatro caracteres de espaço consecutivos. Os caracteres backspace, linefeed e escape geram um aviso de aviso, mas não são processados de outra forma.

O procedimento de janela executa os movimentos de a tecla left, right, end e home caret ao processar a mensagem **WM _ KEYDOWN.** Durante o processamento da ação da tecla SETA PARA A DIREITA, o procedimento de janela verifica o estado da tecla SHIFT e, se ela estiver inobada, seleciona o caractere à direita do a tecla caret à medida que o foco é movido.

Observe que o código a seguir é escrito para que ele possa ser compilado como Unicode ou COMO ANSI. Se o código-fonte definir UNICODE, as cadeias de caracteres serão tratadas como caracteres Unicode; caso contrário, eles serão tratados como caracteres ANSI.

```
#define BUFSIZE 65535
#define SHIFTED 0x8000

LONG APIENTRY MainWndProc(HWND hwndMain, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    HDC hdc;                    // handle to device context
    TEXTMETRIC tm;              // structure for text metrics
    static DWORD dwCharX;       // average width of characters
    static DWORD dwCharY;       // height of characters
    static DWORD dwClientX;     // width of client area
    static DWORD dwClientY;     // height of client area
    static DWORD dwLineLen;     // line length
    static DWORD dwLines;       // text lines in client area
    static int nCaretPosX = 0; // horizontal position of caret
    static int nCaretPosY = 0; // vertical position of caret
    static int nCharWidth = 0; // width of a character
    static int cch = 0;         // characters in buffer
    static int nCurChar = 0;    // index of current character
    static PTCHAR pchInputBuf; // input buffer
    int i, j;                   // loop counters
    int cCR = 0;                // count of carriage returns
    int nCRIndex = 0;           // index of last carriage return
    int nVirtKey;               // virtual-key code
    TCHAR szBuf[128];           // temporary buffer
    TCHAR ch;                   // current character
    PAINTSTRUCT ps;             // required by BeginPaint
    RECT rc;                    // output rectangle for DrawText
    SIZE sz;                    // string dimensions
    COLORREF crPrevText;        // previous text color
    COLORREF crPrevBk;          // previous background color
    size_t * pcch;
    HRESULT hResult;

    switch (uMsg)
    {
        case WM_CREATE:

            // Get the metrics of the current font.

            hdc = GetDC(hwndMain);
            GetTextMetrics(hdc, &tm);
            ReleaseDC(hwndMain, hdc);

            // Save the average character width and height.

            dwCharX = tm.tmAveCharWidth;
            dwCharY = tm.tmHeight;
```

```
            // Allocate a buffer to store keyboard input.

            pchInputBuf = (LPTSTR) GlobalAlloc(GPTR,
                BUFSIZE * sizeof(TCHAR));
            return 0;

        case WM_SIZE:

            // Save the new width and height of the client area.

            dwClientX = LOWORD(lParam);
            dwClientY = HIWORD(lParam);

            // Calculate the maximum width of a line and the
            // maximum number of lines in the client area.

            dwLineLen = dwClientX - dwCharX;
            dwLines = dwClientY / dwCharY;
            break;


        case WM_SETFOCUS:

            // Create, position, and display the caret when the
            // window receives the keyboard focus.

            CreateCaret(hwndMain, (HBITMAP) 1, 0, dwCharY);
            SetCaretPos(nCaretPosX, nCaretPosY * dwCharY);
            ShowCaret(hwndMain);
            break;

        case WM_KILLFOCUS:

            // Hide and destroy the caret when the window loses the
            // keyboard focus.

            HideCaret(hwndMain);
            DestroyCaret();
            break;

        case WM_CHAR:
        // check if current location is close enough to the
        // end of the buffer that a buffer overflow may
        // occur. If so, add null and display contents.
    if (cch > BUFSIZE-5)
    {
        pchInputBuf[cch] = 0x00;
        SendMessage(hwndMain, WM_PAINT, 0, 0);
    }
            switch (wParam)
            {
                case 0x08:  // backspace
                case 0x0A:  // linefeed
                case 0x1B:  // escape
                    MessageBeep((UINT) -1);
                    return 0;

                case 0x09:  // tab

                    // Convert tabs to four consecutive spaces.

                    for (i = 0; i < 4; i++)
                        SendMessage(hwndMain, WM_CHAR, 0x20, 0);
                    return 0;

                case 0x0D:  // carriage return

                    // Record the carriage return and position the
```

```
                // caret at the beginning of the new line.

                pchInputBuf[cch++] = 0x0D;
                nCaretPosX = 0;
                nCaretPosY += 1;
                break;

            default:    // displayable character

                ch = (TCHAR) wParam;
                HideCaret(hwndMain);

                // Retrieve the character's width and output
                // the character.

                hdc = GetDC(hwndMain);
                GetCharWidth32(hdc, (UINT) wParam, (UINT) wParam,
                    &nCharWidth);
                TextOut(hdc, nCaretPosX, nCaretPosY * dwCharY,
                    &ch, 1);
                ReleaseDC(hwndMain, hdc);

                // Store the character in the buffer.

                pchInputBuf[cch++] = ch;

                // Calculate the new horizontal position of the
                // caret. If the position exceeds the maximum,
                // insert a carriage return and move the caret
                // to the beginning of the next line.

                nCaretPosX += nCharWidth;
                if ((DWORD) nCaretPosX > dwLineLen)
                {
                    nCaretPosX = 0;
                    pchInputBuf[cch++] = 0x0D;
                    ++nCaretPosY;
                }
                nCurChar = cch;
                ShowCaret(hwndMain);
                break;
        }
        SetCaretPos(nCaretPosX, nCaretPosY * dwCharY);
        break;

    case WM_KEYDOWN:
        switch (wParam)
        {
            case VK_LEFT:   // LEFT ARROW

                // The caret can move only to the beginning of
                // the current line.

                if (nCaretPosX > 0)
                {
                    HideCaret(hwndMain);

                    // Retrieve the character to the left of
                    // the caret, calculate the character's
                    // width, then subtract the width from the
                    // current horizontal position of the caret
                    // to obtain the new position.

                    ch = pchInputBuf[--nCurChar];
                    hdc = GetDC(hwndMain);
                    GetCharWidth32(hdc, ch, ch, &nCharWidth);
                    ReleaseDC(hwndMain, hdc);
                    nCaretPosX = max(nCaretPosX - nCharWidth,
                        0);
```

```
                0);
            ShowCaret(hwndMain);
        }
        break;

    case VK_RIGHT:  // RIGHT ARROW

        // Caret moves to the right or, when a carriage
        // return is encountered, to the beginning of
        // the next line.

        if (nCurChar < cch)
        {
            HideCaret(hwndMain);

            // Retrieve the character to the right of
            // the caret. If it's a carriage return,
            // position the caret at the beginning of
            // the next line.

            ch = pchInputBuf[nCurChar];
            if (ch == 0x0D)
            {
                nCaretPosX = 0;
                nCaretPosY++;
            }

            // If the character isn't a carriage
            // return, check to see whether the SHIFT
            // key is down. If it is, invert the text
            // colors and output the character.

            else
            {
                hdc = GetDC(hwndMain);
                nVirtKey = GetKeyState(VK_SHIFT);
                if (nVirtKey & SHIFTED)
                {
                    crPrevText = SetTextColor(hdc,
                        RGB(255, 255, 255));
                    crPrevBk = SetBkColor(hdc,
                        RGB(0,0,0));
                    TextOut(hdc, nCaretPosX,
                        nCaretPosY * dwCharY,
                        &ch, 1);
                    SetTextColor(hdc, crPrevText);
                    SetBkColor(hdc, crPrevBk);
                }

                // Get the width of the character and
                // calculate the new horizontal
                // position of the caret.

                GetCharWidth32(hdc, ch, ch, &nCharWidth);
                ReleaseDC(hwndMain, hdc);
                nCaretPosX = nCaretPosX + nCharWidth;
            }
            nCurChar++;
            ShowCaret(hwndMain);
            break;
        }
        break;

    case VK_UP:     // UP ARROW
    case VK_DOWN:   // DOWN ARROW
        MessageBeep((UINT) -1);
        return 0;

    case VK_HOME:   // HOME
```

```
                // Set the caret's position to the upper left
                // corner of the client area.

                nCaretPosX = nCaretPosY = 0;
                nCurChar = 0;
                break;

            case VK_END:    // END

                // Move the caret to the end of the text.

                for (i=0; i < cch; i++)
                {
                    // Count the carriage returns and save the
                    // index of the last one.

                    if (pchInputBuf[i] == 0x0D)
                    {
                        cCR++;
                        nCRIndex = i + 1;
                    }
                }
                nCaretPosY = cCR;

                // Copy all text between the last carriage
                // return and the end of the keyboard input
                // buffer to a temporary buffer.

                for (i = nCRIndex, j = 0; i < cch; i++, j++)
                    szBuf[j] = pchInputBuf[i];
                szBuf[j] = TEXT('\0');

                // Retrieve the text extent and use it
                // to set the horizontal position of the
                // caret.

                hdc = GetDC(hwndMain);
                hResult = StringCchLength(szBuf, 128, pcch);
                if (FAILED(hResult))
                {
                // TODO: write error handler
                }
                GetTextExtentPoint32(hdc, szBuf, *pcch,
                    &sz);
                nCaretPosX = sz.cx;
                ReleaseDC(hwndMain, hdc);
                nCurChar = cch;
                break;

            default:
                break;
        }
        SetCaretPos(nCaretPosX, nCaretPosY * dwCharY);
        break;

    case WM_PAINT:
        if (cch == 0)       // nothing in input buffer
            break;

        hdc = BeginPaint(hwndMain, &ps);
        HideCaret(hwndMain);

        // Set the clipping rectangle, and then draw the text
        // into it.

        SetRect(&rc, 0, 0, dwLineLen, dwClientY);
        DrawText(hdc, pchInputBuf, -1, &rc, DT_LEFT);
```

```
            ShowCaret(hwndMain);
            EndPaint(hwndMain, &ps);
            break;

        // Process other messages.

        case WM_DESTROY:
            PostQuitMessage(0);

            // Free the input buffer.

            GlobalFree((HGLOBAL) pchInputBuf);
            UnregisterHotKey(hwndMain, 0xAAAA);
            break;

        default:
            return DefWindowProc(hwndMain, uMsg, wParam, lParam);
    }
    return NULL;
}
```

Referência de entrada do teclado

15/04/2022 • 2 minutes to read

Nesta seção

- Funções de entrada de teclado
- Mensagens de entrada do teclado
- Notificações de entrada de teclado
- Estruturas de entrada do teclado
- Constantes de entrada de teclado

Funções de entrada de teclado

15/04/2022 • 2 minutes to read

Nesta seção

- ActivateKeyboardLayout
- BlockInput
- Enablewindow
- GetActiveWindow
- Getasynckeystate
- GetFocus
- GetKBCodePage
- GetKeyboardLayout
- GetKeyboardLayoutList
- GetKeyboardLayoutName
- GetKeyboardState
- GetKeyboardType
- GetKeyNameText
- Getkeystate
- GetLastInputInfo
- IsWindowEnabled
- evento _ keybd
- LoadKeyboardLayout
- MapVirtualKey
- MapVirtualKeyEx
- OemKeyScan
- Registerhotkey
- Sendinput
- Setactivewindow
- SetFocus
- SetKeyboardState
- ToAscii
- ToAsciiEx
- ToUnicode
- ToUnicodeEx
- UnloadKeyboardLayout
- UnregisterHotKey
- VkKeyScan
- VkKeyScanEx

Mensagens de entrada do teclado

15/04/2022 • 2 minutes to read

Nesta seção

- WM _
- WM _ SETtecla de atalho

Mensagem WM _ GETHOTKEY

15/04/2022 • 2 minutes to read

Enviado para determinar a tecla quente associada a uma janela.

```
#define WM_GETHOTKEY                    0x0033
```

Parâmetros

wParam

Não usado; deve ser zero.

lParam

Não usado; deve ser zero.

Retornar valor

O valor de retorno é o código de chave virtual e os modificadores para a chave de acesso ou **NULL** se nenhuma tecla de acesso estiver associada à janela. O código de chave virtual está no byte baixo do valor de retorno e os modificadores estão no byte alto. Os modificadores podem ser uma combinação dos seguintes sinalizadores de CommCtrl.h.

VALOR/CÓDIGO DE RETORNO	DESCRIÇÃO
HOTKEYF _ Alt 0x04	tecla ALT
HOTKEYF _ CONTROLE 0x02	Tecla CTRL
HOTKEYF _ EXT 0x08	Chave estendida
HOTKEYF _ SHIFT 0x01	Tecla SHIFT

Comentários

Essas teclas de acesso não estão relacionadas às teclas de acesso definidas **pela função RegisterHotKey.**

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (incluir Windows.h)

Confira também

Referência

Registerhotkey

WM _ SETHOTKEY

Conceitual

Entrada do teclado

_Mensagem de SETtecla do WM

15/04/2022 • 2 minutes to read

Enviado a uma janela para associar uma tecla de acesso à janela. Quando o usuário pressiona a tecla de acesso, o sistema ativa a janela.

```
#define WM_SETHOTKEY                    0x0032
```

Parâmetros

wParam

A palavra de ordem inferior Especifica o código de chave virtual a ser associado à janela.

A palavra de ordem superior pode ser um ou mais dos seguintes valores de CommCtrl. h.

A definição de *wParam* como **NULL** remove a tecla de acesso associada a uma janela.

VALOR	SIGNIFICADO
HOTKEYF _ ALT 0x04	tecla ALT
HOTKEYF _ CONTROLE 0x02	Tecla CTRL
HOTKEYF _ 0x08 ext.	Chave estendida
HOTKEYF _ SHIFT 0x01	Tecla SHIFT

lParam

Este parâmetro não é usado.

Retornar valor

O valor de retorno é um dos seguintes.

VALOR RETORNADO	DESCRIÇÃO
-1	A função não é bem-sucedida; a tecla de acesso é inválida.

VALOR RETORNADO	DESCRIÇÃO
0	A função não é bem-sucedida; a janela é inválida.
1	A função é bem-sucedida e nenhuma outra janela tem a mesma tecla de atalho.
2	A função foi bem-sucedida, mas outra janela já tem a mesma tecla de atalho.

Comentários

Uma tecla de acesso não pode ser associada a uma janela filho.

VK _ ESCAPE, **_ espaço VK** e **_ guia VK** são teclas de acesso inválidas.

Quando o usuário pressiona a tecla de atalho, o sistema gera uma mensagem do **WM _ SYSCOMMAND** com *wParam* igual a **sc _ teclaize** e *lParam* igual ao identificador da janela. Se essa mensagem for passada para **DefWindowProc**, o sistema trará o último Popup ativo da janela (se existir) ou a própria janela (se não houver nenhuma janela pop-up) para o primeiro plano.

Uma janela pode ter apenas uma tecla de acesso. Se a janela já tiver uma tecla de acesso associada a ela, a nova tecla de acesso substituirá a antiga. Se mais de uma janela tiver a mesma tecla de atalho, a janela ativada pela tecla de acesso será aleatória.

Essas teclas de acesso não estão relacionadas às teclas de acesso definidas por **RegisterHotKey**.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser. h (incluir Windows. h)

Confira também

Referência

RegisterHotKey

WM _

SYSCOMMAND do WM _

Conceitual

Entrada de teclado

Notificações de entrada do teclado

15/04/2022 • 2 minutes to read

Nesta seção

- WM _ ACTIVATE
- WM _ APPCOMMAND
- WM _ CHAR
- WM _ DEADCHAR
- WM _ HOTKEY
- WM _ KEYDOWN
- WM _ KEYUP
- WM _ KILLFOCUS
- WM _ SETFOCUS
- WM _ SYSDEADCHAR
- WM _ SYSKEYDOWN
- WM _ SYSKEYUP
- WM _ UNICHAR

Mensagem WM _ ACTIVATE

15/04/2022 • 2 minutes to read

Enviado para a janela que está sendo ativada e a janela que está sendo desativada. Se as janelas usarem a mesma fila de entrada, a mensagem será enviada de forma síncrona, primeiro para o procedimento de janela da janela de nível superior que está sendo desativada e, em seguida, para o procedimento de janela da janela de nível superior que está sendo ativada. Se as janelas usarem filas de entrada diferentes, a mensagem será enviada de forma assíncrona, portanto, a janela será ativada imediatamente.

```
#define WM_ACTIVATE                     0x0006
```

Parâmetros

wParam

A palavra de ordem baixa especifica se a janela está sendo ativada ou desativada. Esse parâmetro pode usar um dos valores a seguir. A palavra de ordem alta especifica o estado minimizado da janela que está sendo ativada ou desativada. Um valor não zero indica que a janela é minimizada.

VALOR	SIGNIFICADO
WA _ ATIVO 1	Ativado por algum método diferente de um clique do mouse (por exemplo, por uma chamada para a função SetActiveWindow ou pelo uso da interface de teclado para selecionar a janela).
WA _ CLICKACTIVE 2	Ativado com um clique do mouse.
WA _ INATIVO 0	Desativado.

lParam

Um alça para a janela que está sendo ativada ou desativada, dependendo do valor do *parâmetro wParam.* Se a palavra de ordem baixa de *wParam* for **WA _ INACTIVE,** *lParam* será o handle para a janela que está sendo ativada. Se a palavra de ordem baixa de *wParam* for **WA _ ACTIVE** ou **WA _ CLICKACTIVE,** *lParam* será o handle para a janela que está sendo desativada. Esse alça pode ser **NULL.**

Retornar valor

Se um aplicativo processa essa mensagem, ele deve retornar zero.

Comentários

Se a janela estiver sendo ativada e não estiver minimizada, a **função DefWindowProc** define o foco do teclado para a janela. Se a janela for ativada com um clique do mouse, ela também receberá uma mensagem **WM _ MOUSEACTIVATE.**

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (incluir Windows.h)

Confira também

Referência

Defwindowproc

Setactivewindow

WM _ MOUSEACTIVATE

WM _ NCACTIVATE

Conceitual

Entrada do teclado

Mensagem _ WM APPCOMMAND

15/04/2022 • 6 minutes to read

Notifica uma janela de que o usuário gerou um evento de comando de aplicativo, por exemplo, clicando em um botão de comando do aplicativo usando o mouse ou digitando uma tecla de comando do aplicativo no teclado.

```
#define WM_APPCOMMAND                   0x0319
```

Parâmetros

wParam

Um alça para a janela em que o usuário clicou no botão ou pressionou a tecla. Essa pode ser uma janela filho da janela que recebe a mensagem. Para obter mais informações sobre como processar essa mensagem, consulte a seção Comentários.

lParam

Use o código a seguir para obter as informações contidas no *parâmetro lParam.*

```
cmd  = GET_APPCOMMAND_LPARAM(lParam);

uDevice = GET_DEVICE_LPARAM(lParam);

dwKeys = GET_KEYSTATE_LPARAM(lParam);
```

O comando do aplicativo *é cmd*, que pode ser um dos valores a seguir.

VALOR	SIGNIFICADO
APPCOMMAND _ BOOST _ DO BOOST 20	A alternar o aumento de bateria para cima e para baixo.
APPCOMMAND _ DOWN _ DO IRON 19	Diminua oixo.
APPCOMMAND _ UP _ DO UP 21 DO IRON	Aumente oixo.
APPCOMMAND _ NAVEGADOR _ PARA TRÁS 1	Navegue para trás.
APPCOMMAND _ FAVORITOS _ DO NAVEGADOR 6	Abra favoritos.

VALOR	SIGNIFICADO
APPCOMMAND _ BROWSER _ FORWARD 2	Navegue para frente.
APPCOMMAND _ PÁGINA _ 7 DO NAVEGADOR	Navegue para casa.
APPCOMMAND _ ATUALIZAÇÃO _ DO NAVEGADOR 3	Página Atualizar.
APPCOMMAND _ PESQUISA _ DE NAVEGADOR 5	Abra a pesquisa.
APPCOMMAND _ BROWSER _ STOP 4	Pare o download.
APPCOMMAND _ CLOSE 31	Feche a janela (não o aplicativo).
APPCOMMAND _ COPY 36	Copie a seleção.
APPCOMMAND _ LISTA _ DE CORREÇÕES 45	Abrirá a lista de correção quando uma palavra for identificada incorretamente durante a entrada de fala.
APPCOMMAND _ CUT 37	Corte a seleção.
APPCOMMAND _ DITAR _ OU _ ALTERNAR _ O CONTROLE _ DE COMANDO 43	Alterna entre dois modos de entrada de fala: ditado e comando/controle (dando comandos a um aplicativo ou acessando menus).
APPCOMMAND _ FIND 28	Abra a **caixa de diálogo** Encontrar.
APPCOMMAND _ FORWARD _ MAIL 40	Encaminhar uma mensagem de email.

VALOR	SIGNIFICADO
APPCOMMAND _ AJUDA 27	Abra a caixa **de diálogo** Ajuda.
APPCOMMAND _ INICIAR _ APP1 17	Inicie o App1.
APPCOMMAND _ INICIAR _ APP2 18	Inicie o App2.
APPCOMMAND _ INICIAR _ O EMAIL 15	Abra o email.
APPCOMMAND _ INICIAR _ MÍDIA _ SELECIONE 16	Vá para o modo Seleção de Mídia.
APPCOMMAND _ MEDIA _ CHANNEL _ DOWN 52	Decrementar o valor do canal, por exemplo, para uma TV ou um ajuste de rádio.
APPCOMMAND _ CANAL _ DE _ MÍDIA ATÉ 51	Incremente o valor do canal, por exemplo, para uma TV ou um ajuste de rádio.
APPCOMMAND _ MEDIA _ FAST _ FORWARD 49	Aumente a velocidade de reprodução de fluxo. Isso pode ser implementado de várias maneiras, por exemplo, usando uma velocidade fixa ou uma rotação por meio de uma série de velocidades crescentes.
APPCOMMAND _ MEDIA _ NEXTTRACK 11	Vá para a próxima faixa.
APPCOMMAND _ MEDIA _ PAUSE 47	Pausa. Se já estiver em pausa, não tome mais nenhuma ação. Este é um comando PAUSE direto que não tem nenhum estado. Se houver botões Reproduzir e Pausar discretos, os aplicativos deverão tomar medidas nesse comando, bem como **APPCOMMAND _ MEDIA _ PLAY _ PAUSE**.
APPCOMMAND _ MEDIA _ PLAY 46	Comece a tocar na posição atual. Se já estiver em pausa, ele será retomado. Este é um comando PLAY direto que não tem nenhum estado. Se houver botões **Reproduzir** e **Pausar** discretos, os aplicativos deverão tomar medidas nesse comando, bem como **APPCOMMAND _ MEDIA PLAY _ _ PAUSE**.

VALOR	SIGNIFICADO
APPCOMMAND _ MEDIA _ PLAY _ PAUSE 14	Reproduzir ou pausar a reprodução. Se houver botões **Reproduzir** e Pausar discretos, os aplicativos deverão tomar medidas nesse comando, bem como **APPCOMMAND _ MEDIA _ PLAY** e **APPCOMMAND _ MEDIA _ PAUSE**.
APPCOMMAND _ MEDIA _ PREVIOUSTRACK 12	Vá para a faixa anterior.
APPCOMMAND _ MEDIA _ RECORD 48	Comece a gravar o fluxo atual.
APPCOMMAND _ MEDIA _ REWIND 50	Volte para trás em um fluxo a uma taxa mais alta de velocidade. Isso pode ser implementado de várias maneiras, por exemplo, usando uma velocidade fixa ou uma rotação por meio de uma série de velocidades crescentes.
APPCOMMAND _ MEDIA _ STOP 13	Pare a reprodução.
APPCOMMAND _ MICROFONE _ _ DESLIGADO/ _ ALTERNÂNCIA 44	Alterne o microfone.
APPCOMMAND _ VOLUME _ DO MICROFONE PARA _ BAIXO 25	Diminua o volume do microfone.
APPCOMMAND _ MUTE _ _ DE VOLUME DO MICROFONE 24	Aumente o microfone.
APPCOMMAND _ VOLUME _ DE MICROFONE PARA _ CIMA 26	Aumente o volume do microfone.
APPCOMMAND _ NOVO 29	Crie uma nova janela.
APPCOMMAND _ OPEN 30	Abra uma janela.

VALOR	SIGNIFICADO
APPCOMMAND _ PASTE 38	Colar
APPCOMMAND _ PRINT 33	Imprimir documento atual.
APPCOMMAND _ REDO 35	Refazer a última ação.
APPCOMMAND _ RESPONDER _ AO _ EMAIL 39	Responder a uma mensagem de email.
APPCOMMAND _ SALVAR 32	Salve o documento atual.
APPCOMMAND _ ENVIAR _ EMAIL 41	Envie uma mensagem de email.
APPCOMMAND _ VERIFICAÇÃO _ ORT SPELL 42	Inicie uma verificação ort ort em .
APPCOMMAND _ TREBLE _ DOWN 22	Diminua o treble.
APPCOMMAND _ TREBLE _ UP 23	Aumente o treble.
APPCOMMAND _ UNDO 34	Desfazer a última ação.
APPCOMMAND _ VOLUME _ PARA BAIXO 9	Reduza o volume.
APPCOMMAND _ VOLUME _ MUTE 8	Mute the volume.
APPCOMMAND _ VOLUME _ ACIMA DE 10	Aumente o volume.

O *componente uDevice* indica o dispositivo de entrada que gerou o evento de entrada e pode ser um dos valores a seguir.

VALOR	SIGNIFICADO
FAPPCOMMAND _ CHAVE 0	O usuário pressionou uma tecla.
FAPPCOMMAND _ MOUSE 0x8000	O usuário clicou em um botão do mouse.
FAPPCOMMAND _ OEM 0x1000	Uma fonte de hardware não identificada gerou o evento. Pode ser um mouse ou um evento de teclado.

O *componente dwKeys* indica se várias chaves virtuais estão inotivas e podem ser um ou mais dos valores a seguir.

VALOR	SIGNIFICADO
MK _ CONTROLE 0x0008	A tecla CTRL está inocizada.
MK _ LBUTTON 0x0001	O botão esquerdo do mouse está ino mouse.
MK _ MBUTTON 0x0010	O botão do meio do mouse está ino mouse.
MK _ RBUTTON 0x0002	O botão direito do mouse está ino mouse.
MK _ SHIFT 0x0004	A tecla SHIFT está inobada.
MK _ XBUTTON1 0x0020	O primeiro botão X está ino mouse.
MK _ XBUTTON2 0x0040	O segundo botão X está ino mouse.

Retornar valor

Se um aplicativo processa essa mensagem, ele deve retornar **TRUE.** Para obter mais informações sobre como

processar o valor de retorno, consulte a seção Comentários.

Comentários

DefWindowProc gera a mensagem **WM _ APPCOMMAND** quando processa a mensagem **WM _ XBUTTONUP** ou **WM _ NCXBUTTONUP** ou quando o usuário digita uma chave de comando do aplicativo.

Se uma janela filho não processar essa mensagem e, em vez disso, chamar **DefWindowProc,** **DefWindowProc** enviará a mensagem para sua janela pai. Se uma janela de nível superior não processar essa mensagem e, em vez disso, chamar **DefWindowProc, DefWindowProc** chamará um gancho de shell com o código de gancho igual a **HSHELL _ APPCOMMAND**.

Para obter as coordenadas do cursor se a mensagem tiver sido gerada por um clique do mouse, o aplicativo poderá **chamar GetMessagePos.** Um aplicativo pode testar se a mensagem foi gerada pelo mouse verificando se *lParam* contém **FAPPCOMMAND _ MOUSE**.

Ao contrário de outras mensagens do Windows, um aplicativo deverá retornar **TRUE** dessa mensagem se processá-la. Isso permitirá que o software que simula essa mensagem em sistemas Windows anteriores ao Windows 2000 determine se o procedimento de janela processou a mensagem ou chamou **DefWindowProc** para processá-la.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (incluir Windows.h)

Confira também

Referência

Defwindowproc

OBTER _ APPCOMMAND _ LPARAM

OBTER _ _ LPARAM DO DISPOSITIVO

GET _ KEYSTATE _ LPARAM

ShellProc

WM _ XBUTTONUP

WM _ NCXBUTTONUP

Conceitual

Entrada do mouse

Mensagem _ WM CHAR

15/04/2022 • 2 minutes to read

Postado na janela com o foco do teclado quando uma mensagem **WM _ KEYDOWN** é convertida pela **função TranslateMessage.** A **mensagem WM _ CHAR** contém o código de caractere da chave que foi pressionada.

```
#define WM_CHAR                         0x0102
```

Parâmetros

wParam

O código de caractere da chave.

lParam

A contagem de repetição, o código de verificação, o sinalizador de chave estendida, o código de contexto, o sinalizador de estado de chave anterior e o sinalizador de estado de transição, conforme mostrado na tabela a seguir.

BITS	SIGNIFICADO
0-15	A contagem de repetição para a mensagem atual. O valor é o número de vezes que o comutado de tecla é autoreado como resultado do usuário manter a chave. Se o tipo de tecla for mantido por tempo suficiente, várias mensagens serão enviadas. No entanto, a contagem de repetição não é cumulativa.
16-23	O código de verificação. O valor depende do OEM.
24	Indica se a chave é uma tecla estendida, como as teclas ALT e CTRL à direita que aparecem em um teclado aprimorado de 101 ou 102 teclas. O valor será 1 se for uma chave estendida; caso contrário, será 0.
25-28	Reservado; não use.
29	O código de contexto. O valor será 1 se a tecla ALT for mantida pressionada enquanto a tecla é pressionada; caso contrário, o valor será 0.
30	O estado da chave anterior. O valor será 1 se a chave estiver inoperante antes que a mensagem seja enviada ou será 0 se a chave estiver inoperante.
31	O estado de transição. O valor será 1 se a chave estiver sendo liberada ou será 0 se a chave estiver sendo pressionada.

Para obter mais detalhes, consulte Sinalizadores de mensagem de teclas.

Retornar valor

Um aplicativo deverá retornar zero se ele processa essa mensagem.

Exemplo

```
LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch (message)
    {

    // ...

    case WM_CHAR:
        OnKeyPress(wParam);
        break;

    default:
        return DefWindowProc(hwnd, message, wParam, lParam);
    }
    return 0;
}
```

Exemplo de Windows exemplos clássicos GitHub.

Comentários

A **mensagem WM _ CHAR** usa o Formato de Transformação Unicode (UTF)-16.

Não há necessariamente uma correspondência um-para-um entre chaves pressionadas e mensagens de caractere geradas e, portanto, as informações na palavra de ordem alta do parâmetro *lParam* geralmente não são úteis para aplicativos. As informações na palavra de ordem alta se aplica somente à mensagem **WM _ KEYDOWN** mais recente que precede a postagem da mensagem **WM _ CHAR.**

Para teclados de 101 e 102 teclas aprimorados, as teclas estendidas são a ALT direita e as teclas CTRL à direita na seção principal do teclado; as teclas INS, DEL, HOME, END, PAGE UP, PAGE DOWN e seta nos clusters à esquerda do teclado numérico; e as chaves de divisão (/) e ENTER no teclado numérico. Alguns outros teclados podem dar suporte ao bit de chave estendida no *parâmetro lParam.*

A **mensagem WM _ UNICHAR** é a mesma que **WM _ CHAR,** exceto pelo uso de UTF-32. Ele foi projetado para enviar ou postar caracteres Unicode em janelas ANSI e pode manipular caracteres de Plano Suplementar Unicode.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (incluir Windows.h)

Confira também

Referência

Translatemessage

WM _ KEYDOWN

WM _ UNICHAR

Conceitual

Entrada do teclado

Mensagem do WM _ DEADCHAR

15/04/2022 • 2 minutes to read

Postado na janela com o foco do teclado quando uma mensagem do **WM _ KEYUP** é convertida pela função **TranslateMessage** . **WM _ DEADCHAR** especifica um código de caractere gerado por uma chave inativa. Uma chave inativa é uma chave que gera um caractere, como o trema (ponto duplo), que é combinado com outro caractere para formar um caractere composto. Por exemplo, o caractere de trema () é gerado digitando-se a chave inativa para o caractere de trema e, em seguida, digitando a tecla o.

```
#define WM_DEADCHAR                     0x0103
```

Parâmetros

wParam

O código de caractere gerado pela chave inativa.

lParam

A contagem de repetição, o código de verificação, o sinalizador de chave estendida, o código de contexto, o sinalizador de estado de chave anterior e o sinalizador de estado de transição, conforme mostrado na tabela a seguir.

BITS	SIGNIFICADO
0-15	A contagem de repetição para a mensagem atual. O valor é o número de vezes que o pressionamento de tecla é repetido de forma automática como resultado do usuário que mantém a chave. Se a tecla de pressionamento for mantida por tempo suficiente, várias mensagens serão enviadas. No entanto, a contagem de repetição não é cumulativa.
16-23	O código de verificação. O valor depende do OEM.
24	Indica se a chave é uma chave estendida, como as teclas ALT direita e CTRL que aparecem em um teclado avançado de 101 ou 102 teclas. O valor será 1 se for uma chave estendida; caso contrário, será 0.
25-28	Reservado Não use.
29	O código do contexto. O valor será 1 se a tecla ALT for mantida pressionada enquanto a tecla for pressionada; caso contrário, o valor será 0.
30	O estado de chave anterior. O valor será 1 se a chave estiver inoperante antes de a mensagem ser enviada ou for 0 se a chave estiver ativa.

BITS	SIGNIFICADO
31	O estado de transição. O valor será 1 se a chave estiver sendo liberada ou for 0 se a chave estiver sendo pressionada.

Para obter mais detalhes, consulte sinalizadores de mensagem de pressionamento de tecla.

Retornar valor

Um aplicativo deve retornar zero se ele processar essa mensagem.

Comentários

A mensagem do **WM _ DEADCHAR** normalmente é usada por aplicativos para fornecer aos comentários do usuário sobre cada tecla pressionada. Por exemplo, um aplicativo pode exibir o acento na posição do caractere atual sem mover o cursor.

Como não há necessariamente uma correspondência de um para um entre as chaves pressionadas e as mensagens de caracteres geradas, as informações na palavra de ordem superior do parâmetro *lParam* geralmente não são úteis para os aplicativos. As informações na palavra de ordem superior aplicam-se apenas à mensagem do **WM _ KEYDOWN** mais recente que precede o lançamento da mensagem do **WM _ DEADCHAR** .

Para teclados avançados de 101 e 102 teclas, as chaves estendidas são as teclas ALT direita e direita na seção principal do teclado; as teclas INS, DEL, HOME, END, PAGE UP, PAGE DOWN e Arrow nos clusters à esquerda do teclado numérico; e a divisão (/) e inserir chaves no teclado numérico. Alguns teclados podem dar suporte ao bit de chave estendida no parâmetro *lParam* .

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser. h (incluir Windows. h)

Confira também

Referência

TranslateMessage

o WM _ KEYDOWN

o WM _ KEYUP

SYSDEADCHAR do WM _

Conceitual

Entrada de teclado

Mensagem de tecla de atalho do WM _

15/04/2022 • 2 minutes to read

Postado quando o usuário pressiona uma tecla de atalho registrada pela função **RegisterHotKey** . A mensagem é colocada na parte superior da fila de mensagens associada ao thread que registrou a tecla de acesso.

```
#define WM_HOTKEY                       0x0312
```

Parâmetros

wParam

O identificador da tecla de acesso que gerou a mensagem. Se a mensagem tiver sido gerada por uma tecla de acesso definida pelo sistema, esse parâmetro será um dos valores a seguir.

VALOR	SIGNIFICADO
IDHOT _ SNAPDESKTOP -2	A tecla de atalho "ajustar área de trabalho" foi pressionada.
IDHOT _ SNAPWINDOW -1	A tecla de atalho "ajustar janela" foi pressionada.

lParam

A palavra de ordem inferior Especifica as chaves que foram pressionadas em combinação com a chave especificada pela palavra de ordem superior para gerar a mensagem de **_ tecla de atalho do WM** . Esta palavra pode ser um ou mais dos valores a seguir. A palavra de ordem superior especifica o código de chave virtual da tecla de atalho.

VALOR	SIGNIFICADO
Mod _ ALT 0x0001	A tecla ALT foi mantida inativa.
Mod _ 0X0002 de controle	A tecla CTRL foi mantida inativa.
Mod _ SHIFT 0x0004	A tecla SHIFT foi mantida inativa.

VALOR	SIGNIFICADO
Mod _ WIN 0x0008	A chave do WINDOWS foi mantida. essas chaves são rotuladas com o logotipo Windows. as teclas de tecla que envolvem a chave de Windows são reservadas para uso pelo sistema operacional.

Comentários

WM _ A tecla de atalho não está relacionada às teclas de acesso do WM prekeys e do **WM _ settecla** . _ A mensagem de _ **tecla** de acesso do WM é enviada para chaves ativas genéricas enquanto as mensagens do WM AutoKeys e do **WM _** estão relacionadas às teclas de **acesso de ativação _** do Windows.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser. h (incluir Windows. h)

Confira também

Referência

RegisterHotKey

WM _

WM _ SETtecla de atalho

Conceitual

Entrada de teclado

Mensagem de KEYDOWN do WM _

15/04/2022 • 2 minutes to read

Postado na janela com o foco do teclado quando uma tecla que não é do sistema é pressionada. Uma chave que não seja do sistema é uma chave que é pressionada quando a tecla ALT não é pressionada.

```
#define WM_KEYDOWN                      0x0100
```

Parâmetros

wParam

O código de chave virtual da chave que não é do sistema. Consulte códigos de chave virtual.

lParam

A contagem de repetição, o código de verificação, o sinalizador de chave estendida, o código de contexto, o sinalizador de estado de chave anterior e o sinalizador de estado de transição, conforme mostrado a seguir.

BITS	SIGNIFICADO
0-15	A contagem de repetição para a mensagem atual. O valor é o número de vezes que o pressionamento de tecla é repetido de forma automática como resultado do usuário que mantém a chave. Se a tecla de pressionamento for mantida por tempo suficiente, várias mensagens serão enviadas. No entanto, a contagem de repetição não é cumulativa.
16-23	O código de verificação. O valor depende do OEM.
24	Indica se a chave é uma chave estendida, como as teclas ALT direita e CTRL que aparecem em um teclado avançado de 101 ou 102 teclas. O valor será 1 se for uma chave estendida; caso contrário, será 0.
25-28	Reservado Não use.
29	O código do contexto. O valor é sempre 0 para uma mensagem do **WM _ KEYDOWN** .
30	O estado de chave anterior. O valor será 1 se a chave estiver inoperante antes de a mensagem ser enviada ou for zero se a chave estiver ativa.
31	O estado de transição. O valor é sempre 0 para uma mensagem do **WM _ KEYDOWN** .

Para obter mais detalhes, consulte sinalizadores de mensagem de pressionamento de tecla.

Retornar valor

Um aplicativo deve retornar zero se ele processar essa mensagem.

Exemplo

```
LRESULT CALLBACK HostWndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch (message)
    {
    case WM_KEYDOWN:
        if (wParam == VK_ESCAPE)
        {
            if (isFullScreen)
            {
                GoPartialScreen();
            }
        }
        break;

    // ...
    default:
        return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}
```

exemplo de exemplos clássicos Windows em GitHub.

Comentários

Se a tecla F10 for pressionada, a função **DefWindowProc** definirá um sinalizador interno. Quando **DefWindowProc** recebe a mensagem do **WM _ KEYUP** , a função verifica se o sinalizador interno está definido e, nesse caso, envia uma mensagem do **WM _ SYSCOMMAND** para a janela de nível superior. O parâmetro **WM _ SYSCOMMAND** da mensagem é definido como SC _ keymenu.

Devido ao recurso AutoRepetir, mais de uma mensagem do **WM _ KEYDOWN** pode ser postada antes que uma mensagem do **WM _ KEYUP** seja postada. O estado de chave anterior (bit 30) pode ser usado para determinar se a mensagem do **WM _ KEYDOWN** indica a primeira transição para baixo ou uma transição repetida.

Para teclados avançados de 101 e 102 teclas, as chaves estendidas são as teclas ALT e CTRL pressionadas na seção principal do teclado; as teclas INS, DEL, HOME, END, PAGE UP, PAGE DOWN e Arrow nos clusters à esquerda do teclado numérico; e a divisão (/) e inserir chaves no teclado numérico. Outros teclados podem dar suporte ao bit de chave estendida no parâmetro *lParam* .

Os aplicativos devem passar *wParam* para **TranslateMessage** sem alterá-lo.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]

REQUISITO	VALOR
Cabeçalho	Winuser. h (incluir Windows. h)

Confira também

Referência

DefWindowProc

TranslateMessage

caractere do WM _

o WM _ KEYUP

SYSCOMMAND do WM _

Conceitual

Entrada de teclado

Mensagem WM _ KEYUP

15/04/2022 • 2 minutes to read

Postado na janela com o foco do teclado quando uma tecla não sistema é liberada. Uma tecla não sistema é uma tecla pressionada quando a tecla ALT não é pressionada ou uma tecla de teclado pressionada quando uma janela tem o foco do teclado.

```
#define WM_KEYUP                        0x0101
```

Parâmetros

wParam

O código de chave virtual da chave não sistema. Consulte Códigos de chave virtual.

lParam

A contagem de repetição, o código de verificação, o sinalizador de chave estendida, o código de contexto, o sinalizador de estado de chave anterior e o sinalizador de estado de transição, conforme mostrado na tabela a seguir.

BITS	SIGNIFICADO
0-15	A contagem de repetição para a mensagem atual. O valor é o número de vezes que o comutado de tecla é autoreado como resultado do usuário manter a chave. A contagem de repetição é sempre 1 para uma **mensagem WM _ KEYUP.**
16-23	O código de verificação. O valor depende do OEM.
24	Indica se a chave é uma tecla estendida, como as teclas ALT e CTRL à direita que aparecem em um teclado aprimorado de 101 ou 102 teclas. O valor será 1 se for uma chave estendida; caso contrário, será 0.
25-28	Reservado; não use.
29	O código de contexto. O valor é sempre 0 para uma **mensagem WM _ KEYUP.**
30	O estado da chave anterior. O valor é sempre 1 para uma **mensagem WM _ KEYUP.**
31	O estado de transição. O valor é sempre 1 para uma **mensagem WM _ KEYUP.**

Para obter mais detalhes, consulte Sinalizadores de mensagem de teclas.

Retornar valor

Um aplicativo deverá retornar zero se ele processa essa mensagem.

Comentários

A **função DefWindowProc** enviará uma mensagem **WM _ SYSCOMMAND** para a janela de nível superior se a tecla F10 ou a tecla ALT tiver sido liberada. O *parâmetro wParam* da mensagem é definido como SC _ KEYMENU.

Para teclados de 101 e 102 teclas aprimorados, as teclas estendidas são as teclas ALT e CTRL corretas na seção principal do teclado; as teclas INS, DEL, HOME, END, PAGE UP, PAGE DOWN e seta nos clusters à esquerda do teclado numérico; e as chaves de divisão (/) e ENTER no teclado numérico. Outros teclados podem dar suporte ao bit de chave estendida no *parâmetro lParam.*

Os aplicativos *devem passar wParam* **para TranslateMessage** sem alterá-lo.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (incluir Windows.h)

Confira também

Referência

Defwindowproc

Translatemessage

WM _ KEYDOWN

WM _ SYSCOMMAND

Conceitual

Entrada do teclado

Mensagem do WM _ KILLFOCUS

15/04/2022 • 2 minutes to read

Enviado a uma janela imediatamente antes de perder o foco do teclado.

```
#define WM_KILLFOCUS                    0x0008
```

Parâmetros

wParam

Um identificador para a janela que recebe o foco do teclado. Este parâmetro pode ser **NULL**.

lParam

Este parâmetro não é usado.

Retornar valor

Um aplicativo deve retornar zero se ele processar essa mensagem.

Comentários

Se um aplicativo estiver exibindo um cursor, o cursor deverá ser destruído neste ponto.

Ao processar essa mensagem, não faça nenhuma chamada de função que exiba ou ative uma janela. Isso faz com que o thread gere controle e pode fazer com que o aplicativo pare de responder às mensagens. Para obter mais informações, consulte deadlocks de mensagem.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser. h (incluir Windows. h)

Confira também

Referência

SetFocus

WM _ SETFOCUS

Conceitual

Entrada de teclado

Mensagem do WM _ SETFOCUS

15/04/2022 • 2 minutes to read

Enviado a uma janela depois de ter obtido o foco do teclado.

```
#define WM_SETFOCUS                     0x0007
```

Parâmetros

wParam

Um identificador para a janela que perdeu o foco do teclado. Este parâmetro pode ser **NULL**.

lParam

Este parâmetro não é usado.

Retornar valor

Um aplicativo deve retornar zero se ele processar essa mensagem.

Comentários

Para exibir um cursor, um aplicativo deve chamar as funções de cursor apropriadas quando receber a mensagem do **WM _ SETFOCUS** .

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser. h (incluir Windows. h)

Confira também

Referência

SetFocus

KILLFOCUS do WM _

Conceitual

Entrada de teclado

Mensagem _ WM SYSDEADCHAR

15/04/2022 • 2 minutes to read

Enviado para a janela com o foco do teclado quando uma mensagem **WM _ SYSKEYDOWN** é convertida pela **função TranslateMessage.** **WM _ SYSDEADCHAR** especifica o código de caractere de uma chave morta do sistema, ou seja, uma tecla morta pressionada enquanto mantém pressionada a tecla ALT.

```
#define WM_SYSDEADCHAR                  0x0107
```

Parâmetros

wParam

O código de caractere gerado pela chave morta do sistema, ou seja, uma tecla morta pressionada enquanto mantém pressionada a tecla ALT.

lParam

A contagem de repetição, o código de verificação, o sinalizador de chave estendida, o código de contexto, o sinalizador de estado de chave anterior e o sinalizador de estado de transição, conforme mostrado na tabela a seguir.

BITS	SIGNIFICADO
0-15	A contagem de repetição para a mensagem atual. O valor é o número de vezes que o comutado de tecla é autoreado como resultado do usuário manter a chave. Se o tipo de tecla for mantido por tempo suficiente, várias mensagens serão enviadas. No entanto, a contagem de repetição não é cumulativa.
16-23	O código de verificação. O valor depende do OEM.
24	Indica se a chave é uma tecla estendida, como as teclas ALT e CTRL à direita que aparecem em um teclado aprimorado de 101 ou 102 teclas. O valor será 1 se for uma chave estendida; caso contrário, será 0.
25-28	Reservado; não use.
29	O código de contexto. O valor será 1 se a tecla ALT for mantida pressionada enquanto a tecla é pressionada; caso contrário, o valor será 0.
30	O estado da chave anterior. O valor será 1 se a chave estiver inoperante antes que a mensagem seja enviada ou será 0 se a chave estiver inoperante.
31	Estado de transição. O valor será 1 se a chave estiver sendo liberada ou será 0 se a chave estiver sendo pressionada.

Para obter mais detalhes, consulte Sinalizadores de mensagem de teclas.

Retornar valor

Um aplicativo deverá retornar zero se ele processa essa mensagem.

Comentários

Para teclados de 101 e 102 teclas aprimorados, as teclas estendidas são as teclas ALT e CTRL corretas na seção principal do teclado; as teclas INS, DEL, HOME, END, PAGE UP, PAGE DOWN e seta nos clusters à esquerda do teclado numérico; e as chaves de divisão (/) e ENTER no teclado numérico. Outros teclados podem dar suporte ao bit de chave estendida no *parâmetro lParam.*

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (incluir Windows.h)

Confira também

Referência

Translatemessage

WM _ DEADCHAR

WM _ SYSKEYDOWN

Conceitual

Entrada do teclado

Mensagem do WM _ SYSKEYDOWN

15/04/2022 • 3 minutes to read

Postado na janela com o foco do teclado quando o usuário pressiona a tecla F10 (que ativa a barra de menus) ou mantém a tecla ALT pressionada e, em seguida, pressiona outra tecla. Ele também ocorre quando nenhuma janela tem o foco do teclado no momento; Nesse caso, a mensagem do **WM _ SYSKEYDOWN** é enviada para a janela ativa. A janela que recebe a mensagem pode distinguir entre esses dois contextos verificando o código de contexto no parâmetro *lParam* .

```
#define WM_SYSKEYDOWN                   0x0104
```

Parâmetros

wParam

O código de chave virtual da chave que está sendo pressionada. Consulte **códigos de chave virtual**.

lParam

A contagem de repetição, o código de verificação, o sinalizador de chave estendida, o código de contexto, o sinalizador de estado de chave anterior e o sinalizador de estado de transição, conforme mostrado na tabela a seguir.

BITS	SIGNIFICADO
0-15	A contagem de repetição para a mensagem atual. O valor é o número de vezes que o pressionamento de tecla é repetido de forma automática como resultado do usuário que mantém a chave. Se a tecla de pressionamento for mantida por tempo suficiente, várias mensagens serão enviadas. No entanto, a contagem de repetição não é cumulativa.
16-23	O código de verificação. O valor depende do OEM.
24	Indica se a chave é uma chave estendida, como as teclas ALT direita e CTRL que aparecem em um teclado avançado de 101 ou 102 teclas. O valor será 1 se for uma chave estendida; caso contrário, será 0.
25-28	Reservado Não use.
29	O código do contexto. O valor será 1 se a tecla ALT estiver inoperante enquanto a tecla for pressionada; será 0 se a mensagem do **WM _ SYSKEYDOWN** for postada na janela ativa porque nenhuma janela tem o foco do teclado.
30	O estado de chave anterior. O valor será 1 se a chave estiver inoperante antes de a mensagem ser enviada ou for 0 se a chave estiver ativa.

BITS	SIGNIFICADO
31	O estado de transição. O valor é sempre 0 para uma mensagem do **WM _ SYSKEYDOWN** .

Para obter mais detalhes, consulte sinalizadores de mensagem de pressionamento de tecla.

Retornar valor

Um aplicativo deve retornar zero se ele processar essa mensagem.

Comentários

A função **DefWindowProc** examina a chave especificada e gera uma mensagem do **WM _ SYSCOMMAND** se a chave for Tab ou Enter.

Quando o código de contexto for zero, a mensagem poderá ser passada para a função **TranslateAccelerator** , que o tratará como se fosse uma mensagem de chave normal em vez de uma mensagem de chave de caracteres. Isso permite que as teclas de aceleração sejam usadas com a janela ativa mesmo que a janela ativa não tenha o foco do teclado.

Devido à repetição automática, mais de uma mensagem do **WM _ SYSKEYDOWN** pode ocorrer antes de uma mensagem do **WM _ SYSKEYUP** ser enviada. O estado de chave anterior (bit 30) pode ser usado para determinar se a mensagem do **WM _ SYSKEYDOWN** indica a primeira transição para baixo ou uma transição repetida.

Para teclados avançados de 101 e 102 teclas, as chaves avançadas são as teclas ALT e CTRL na seção principal do teclado; as teclas INS, DEL, HOME, END, PAGE UP, PAGE DOWN e Arrow nos clusters à esquerda do teclado numérico; e a divisão (/) e inserir chaves no teclado numérico. Outros teclados podem dar suporte ao bit de chave estendida no parâmetro *lParam* .

Essa mensagem também é enviada sempre que o usuário pressiona a tecla F10 sem a tecla ALT.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser. h (incluir Windows. h)

Confira também

Referência

DefWindowProc

TranslateAccelerator

SYSCOMMAND do WM _

SYSKEYUP do WM _

Conceitual

Entrada de teclado

Mensagem _ WM SYSKEYUP

15/04/2022 • 3 minutes to read

Postado na janela com o foco do teclado quando o usuário libera uma tecla que foi pressionada enquanto a tecla ALT era pressionada. Isso também ocorre quando nenhuma janela tem o foco do teclado no momento; nesse caso, a mensagem **WM _ SYSKEYUP** é enviada para a janela ativa. A janela que recebe a mensagem pode distinguir entre esses dois contextos verificando o código de contexto no *parâmetro lParam.*

Uma janela recebe essa mensagem por meio de **sua função WindowProc.**

```
#define WM_SYSKEYUP                     0x0105
```

Parâmetros

wParam

O código de chave virtual da chave que está sendo liberada. Consulte **Códigos de chave virtual.**

lParam

A contagem de repetição, o código de verificação, o sinalizador de chave estendida, o código de contexto, o sinalizador de estado de chave anterior e o sinalizador de estado de transição, conforme mostrado na tabela a seguir.

BITS	SIGNIFICADO
0-15	A contagem de repetição para a mensagem atual. O valor é o número de vezes que o comutado de tecla é autoreado como resultado do usuário manter a chave. A contagem de repetição é sempre uma para uma _ **mensagem WM SYSKEYUP.**
16-23	O código de verificação. O valor depende do OEM.
24	Indica se a chave é uma tecla estendida, como as teclas ALT e CTRL à direita que aparecem em um teclado aprimorado de 101 ou 102 teclas. O valor será 1 se for uma chave estendida; caso contrário, será zero.
25-28	Reservado; não use.
29	O código de contexto. O valor será 1 se a tecla ALT estiver inobada enquanto a chave é liberada; será zero se a mensagem **WM _ SYSKEYUP** for postada na janela ativa porque nenhuma janela tem o foco do teclado.
30	O estado da chave anterior. O valor é sempre 1 para uma mensagem _ **WM SYSKEYUP.**
31	O estado de transição. O valor é sempre 1 para uma mensagem _ **WM SYSKEYUP.**

Para obter mais detalhes, consulte Sinalizadores de mensagem de teclas.

Retornar valor

Um aplicativo deverá retornar zero se ele processa essa mensagem.

Comentários

A **função DefWindowProc** enviará uma mensagem **WM _ SYSCOMMAND** para a janela de nível superior se a tecla F10 ou a tecla ALT tiver sido liberada. O *parâmetro wParam* da mensagem é definido como **SC _ KEYMENU**.

Quando o código de contexto é zero, a mensagem pode ser passada para a função **TranslateAccelerator,** que o tratará como se fosse uma mensagem de chave normal em vez de uma mensagem de chave de caractere. Isso permite que as teclas de acelerador sejam usadas com a janela ativa, mesmo que a janela ativa não tenha o foco do teclado.

Para teclados de 101 e 102 teclas aprimorados, as teclas estendidas são as teclas ALT e CTRL corretas na seção principal do teclado; as teclas INS, DEL, HOME, END, PAGE UP, PAGE DOWN e seta nos clusters à esquerda do teclado numérico; e as chaves de divisão (/) e ENTER no teclado numérico. Outros teclados podem dar suporte ao bit de chave estendida no *parâmetro lParam.*

Para não EUA teclados de 102 teclas aprimorados, a tecla ALT direita é tratada como uma tecla CTRL+ALT. A tabela a seguir mostra a sequência de mensagens que resultam quando o usuário pressiona e libera essa chave.

MENSAGEM	CÓDIGO DE CHAVE VIRTUAL
WM _ KEYDOWN	**CONTROLE _ de VK**
WM _ KEYDOWN	**MENU da VK _**
WM _ KEYUP	**CONTROLE _ de VK**
WM _ SYSKEYUP	**MENU da VK _**

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (incluir Windows.h)

Confira também

Referência

Defwindowproc

Translateaccelerator

WM _ SYSCOMMAND

WM _ SYSKEYDOWN

Conceitual

Entrada do teclado

Mensagem do WM _ UNICHAR

15/04/2022 • 3 minutes to read

A mensagem do **WM _ UNICHAR** pode ser usada por um aplicativo para postar a entrada para outras janelas. Essa mensagem contém o código de caractere da chave que foi pressionada. (Teste se um aplicativo de destino pode processar mensagens do **WM _ UNICHAR** enviando a mensagem com *wParam* definido como **Unicode _ nochar**.)

```
#define WM_UNICHAR                  0x0109
```

Parâmetros

wParam

O código de caractere da chave.

Se *wParam* for **Unicode _ nochar** e o aplicativo processar essa mensagem, retornará **true**. A função **DefWindowProc** retornará **false** (o padrão).

Se *wParam* não for **Unicode _ nochar**, retornará **false**. O Unicode **DefWindowProc** posta uma mensagem do **WM _ Char** com os mesmos parâmetros e a função ANSI **DefWindowProc** posta uma ou duas mensagens do **WM _ Char** com os caracteres ANSI correspondentes.

lParam

A contagem de repetição, o código de verificação, o sinalizador de chave estendida, o código de contexto, o sinalizador de estado de chave anterior e o sinalizador de estado de transição, conforme mostrado na tabela a seguir.

BITS	SIGNIFICADO
0-15	A contagem de repetição para a mensagem atual. O valor é o número de vezes que o pressionamento de tecla é repetido de forma automática como resultado do usuário que mantém a chave. Se a tecla de pressionamento for mantida por tempo suficiente, várias mensagens serão enviadas. No entanto, a contagem de repetição não é cumulativa.
16-23	O código de verificação. O valor depende do OEM.
24	Indica se a chave é uma chave estendida, como as teclas ALT direita e CTRL que aparecem em um teclado avançado de 101 ou 102 teclas. O valor será 1 se for uma chave estendida; caso contrário, será 0.
25-28	Reservado Não use.
29	O código do contexto. O valor será 1 se a tecla ALT for mantida pressionada enquanto a tecla for pressionada; caso contrário, o valor será 0.

BITS	SIGNIFICADO
30	O estado de chave anterior. O valor será 1 se a chave estiver inoperante antes de a mensagem ser enviada ou for 0 se a chave estiver ativa.
31	O estado de transição. O valor será 1 se a chave estiver sendo liberada ou for 0 se a chave estiver sendo pressionada.

Para obter mais detalhes, consulte sinalizadores de mensagem de pressionamento de tecla.

Retornar valor

Um aplicativo deve retornar zero se ele processar essa mensagem.

Comentários

A mensagem do **WM _ UNICHAR** é semelhante ao **WM _ Char**, mas usa o formato UTF (Unicode transformation Format)-32, enquanto o **WM _ Char** usa UTF-16.

Essa mensagem foi criada para enviar ou postar caracteres Unicode para janelas ANSI e pode manipular caracteres de plano suplementar Unicode.

Como não há necessariamente uma correspondência de um para um entre as chaves pressionadas e as mensagens de caracteres geradas, as informações na palavra de ordem superior do parâmetro *lParam* geralmente não são úteis para os aplicativos. As informações na palavra de ordem superior aplicam-se apenas à mensagem do **WM _ KEYDOWN** mais recente que precede o lançamento da mensagem do **WM _ UNICHAR** .

Para teclados avançados de 101 e 102 teclas, as chaves estendidas são as teclas ALT direita e direita na seção principal do teclado; as teclas INS, DEL, HOME, END, PAGE UP, PAGE DOWN e Arrow nos clusters à esquerda do teclado numérico; e a divisão (/) e inserir chaves no teclado numérico. Alguns teclados podem dar suporte ao bit de chave estendida no parâmetro *lParam* .

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows [Somente aplicativos da área de trabalho XP]
Servidor mínimo com suporte	Windows [Somente aplicativos da área de trabalho do servidor 2003]
Cabeçalho	Winuser. h (incluir Windows. h)

Confira também

Referência

TranslateMessage

caractere do WM _

o WM _ KEYDOWN

Conceitual

Entrada de teclado

Estruturas de entrada do teclado

15/04/2022 • 2 minutes to read

Nesta seção

- HARDWAREINPUT
- ENTRADA
- KEYBDINPUT
- LASTINPUTINFO
- MOUSEINPUT

Constantes de entrada de teclado

15/04/2022 • 2 minutes to read

- Códigos de chave virtual

Virtual-Key código

15/04/2022 • 5 minutes to read

A tabela a seguir mostra os nomes de constante simbólicos, valores hexadecimais e equivalentes de mouse ou teclado para os códigos de chave virtual usados pelo sistema. Os códigos são listados em ordem numérica.

CONSTANTE	VALOR	DESCRIÇÃO
`VK_LBUTTON`	0x01	Botão do mouse esquerdo
`VK_RBUTTON`	0x02	Botão direito do mouse
`VK_CANCEL`	0x03	Processamento de quebra de controle
`VK_MBUTTON`	0x04	Botão do meio do mouse (mouse de três botões)
`VK_XBUTTON1`	0x05	Botão do mouse X1
`VK_XBUTTON2`	0x06	Botão do mouse X2
`-`	0x07	Indefinido
`VK_BACK`	0x08	Chave BACKSPACE
`VK_TAB`	0x09	tecla TAB
`-`	0x0A-0B	Reservado
`VK_CLEAR`	0x0C	Chave CLEAR
`VK_RETURN`	0x0D	Tecla ENTER
`-`	0x0E-0F	Indefinido
`VK_SHIFT`	0x10	Tecla SHIFT
`VK_CONTROL`	0x11	Tecla CTRL
`VK_MENU`	0x12	tecla ALT
`VK_PAUSE`	0x13	Tecla PAUSE
`VK_CAPITAL`	0x14	CAPS LOCK chave
`VK_KANA`	0x15	Modo Kana do IME

CONSTANTE	VALOR	DESCRIÇÃO
`VK_HANGUEL`	0x15	Modo hanguel do IME (mantido para compatibilidade; use `VK_HANGUL`)
`VK_HANGUL`	0x15	Modo Hangul do IME
`VK_IME_ON`	0x16	IME On
`VK_JUNJA`	0x17	Modo Junja do IME
`VK_FINAL`	0x18	Modo final do IME
`VK_HANJA`	0x19	Modo Hanja do IME
`VK_KANJI`	0x19	Modo Kanji do IME
`VK_IME_OFF`	0x1A	IME Off
`VK_ESCAPE`	0x1B	Chave ESC
`VK_CONVERT`	0x1C	Conversão de IME
`VK_NONCONVERT`	0x1D	IME nãoconvertido
`VK_ACCEPT`	0x1E	Aceitar IME
`VK_MODECHANGE`	0x1F	Solicitação de alteração do modo IME
`VK_SPACE`	0x20	BARRA DE ESPAÇOS
`VK_PRIOR`	0x21	Tecla PAGE UP
`VK_NEXT`	0x22	Tecla PAGE DOWN
`VK_END`	0x23	Tecla END
`VK_HOME`	0x24	Tecla HOME
`VK_LEFT`	0x25	Tecla DE SETA PARA A ESQUERDA
`VK_UP`	0x26	Tecla DE SETA PARA CIMA
`VK_RIGHT`	0x27	Tecla DE SETA PARA A DIREITA
`VK_DOWN`	0x28	Tecla DE SETA PARA BAIXO
`VK_SELECT`	0x29	Tecla SELECT
`VK_PRINT`	0x2A	Tecla PRINT

CONSTANTE	VALOR	DESCRIÇÃO
`VK_EXECUTE`	0x2B	Chave EXECUTE
`VK_SNAPSHOT`	0x2C	Tecla PRINT SCREEN
`VK_INSERT`	0x2D	Chave INS
`VK_DELETE`	0x2E	Tecla DEL
`VK_HELP`	0x2F	Chave DE AJUDA
	0x30	0 chave
	0x31	1 chave
	0x32	2 chaves
	0x33	3 chaves
	0x34	4 chaves
	0x35	5 chaves
	0x36	6 chaves
	0x37	7 chaves
	0x38	8 chaves
	0x39	9 chaves
`-`	0x3A-40	Indefinido
	0x41	Uma chave
	0x42	Chave B
	0x43	Chave C
	0x44	Chave D
	0x45	Chave E
	0x46	Tecla F
	0x47	Chave G
	0x48	Tecla H
	0x49	Chave I

CONSTANTE	VALOR	DESCRIÇÃO
	0x4A	Chave J
	0x4B	Chave K
	0x4C	Chave L
	0x4D	Chave M
	0x4E	N chave
	0x4F	Chave O
	0x50	Chave P
	0x51	Chave Q
	0x52	Tecla R
	0x53	Chave S
	0x54	Chave T
	0x55	Tecla U
	0x56	Chave V
	0x57	Chave W
	0x58	Chave X
	0x59	Chave Y
	0x5A	Tecla Z
`VK_LWIN`	0x5B	Tecla Windows esquerda (teclado natural)
`VK_RWIN`	0x5C	Tecla Windows direita (teclado natural)
`VK_APPS`	0x5D	Chave de aplicativos (teclado natural)
`-`	0x5E	Reservado
`VK_SLEEP`	0x5F	Tecla de sleep do computador
`VK_NUMPAD0`	0x60	Tecla numeric keypad 0
`VK_NUMPAD1`	0x61	Tecla 1 do teclado numérico
`VK_NUMPAD2`	0x62	Tecla numeric keypad 2

CONSTANTE	VALOR	DESCRIÇÃO
`VK_NUMPAD3`	0x63	Tecla numeric keypad 3
`VK_NUMPAD4`	0x64	Tecla 4 do teclado numérico
`VK_NUMPAD5`	0x65	Tecla numeric keypad 5
`VK_NUMPAD6`	0x66	Tecla numeric keypad 6
`VK_NUMPAD7`	0x67	Tecla 7 do teclado numérico
`VK_NUMPAD8`	0x68	Tecla 8 do teclado numérico
`VK_NUMPAD9`	0x69	Tecla 9 do teclado numérico
`VK_MULTIPLY`	0x6A	Multiplicar chave
`VK_ADD`	0x6B	Adicionar chave
`VK_SEPARATOR`	0x6C	Chave do separador
`VK_SUBTRACT`	0x6D	Subtrair chave
`VK_DECIMAL`	0x6E	Chave decimal
`VK_DIVIDE`	0x6F	Dividir chave
`VK_F1`	0x70	Tecla F1
`VK_F2`	0x71	Tecla F2
`VK_F3`	0x72	Tecla F3
`VK_F4`	0x73	Tecla F4
`VK_F5`	0x74	Tecla F5
`VK_F6`	0x75	Tecla F6
`VK_F7`	0x76	Tecla F7
`VK_F8`	0x77	Tecla F8
`VK_F9`	0x78	Tecla F9
`VK_F10`	0x79	Tecla F10
`VK_F11`	0x7A	Tecla F11

CONSTANTE	VALOR	DESCRIÇÃO
`VK_F12`	0x7B	Tecla F12
`VK_F13`	0x7C	Tecla F13
`VK_F14`	0x7D	Tecla F14
`VK_F15`	0x7E	Tecla F15
`VK_F16`	0x7F	Tecla F16
`VK_F17`	0x80	Tecla F17
`VK_F18`	0x81	Tecla F18
`VK_F19`	0x82	Tecla F19
`VK_F20`	0x83	Tecla F20
`VK_F21`	0x84	Tecla F21
`VK_F22`	0x85	Tecla F22
`VK_F23`	0x86	Tecla F23
`VK_F24`	0x87	Tecla F24
`-`	0x88-8F	Não atribuído
`VK_NUMLOCK`	0x90	Chave NUM LOCK
`VK_SCROLL`	0x91	TECLA SCROLL LOCK
	0x92-96	Específico do OEM
`-`	0x97-9F	Não atribuído
`VK_LSHIFT`	0xA0	Tecla SHIFT esquerda
`VK_RSHIFT`	0xA1	Tecla SHIFT para a direita
`VK_LCONTROL`	0xA2	Tecla CONTROL esquerda
`VK_RCONTROL`	0xA3	Tecla CONTROL à direita
`VK_LMENU`	0xA4	Tecla MENU à esquerda
`VK_RMENU`	0xA5	Tecla MENU à direita

CONSTANTE	VALOR	DESCRIÇÃO
VK_BROWSER_BACK	0xA6	Tecla Voltar do Navegador
VK_BROWSER_FORWARD	0xA7	Tecla De encaminhamento do navegador
VK_BROWSER_REFRESH	0xA8	Chave de atualização do navegador
VK_BROWSER_STOP	0xA9	Tecla De parada do navegador
VK_BROWSER_SEARCH	0xAA	Chave de Pesquisa do Navegador
VK_BROWSER_FAVORITES	0xAB	Chave favoritos do navegador
VK_BROWSER_HOME	0xAC	Tecla Iniciar e Página Inicial do Navegador
VK_VOLUME_MUTE	0xAD	Chave de mute de volume
VK_VOLUME_DOWN	0xAE	Tecla Volume Down
VK_VOLUME_UP	0xAF	Tecla Volume Up
VK_MEDIA_NEXT_TRACK	0xB0	Próxima tecla Track
VK_MEDIA_PREV_TRACK	0xB1	Chave de faixa anterior
VK_MEDIA_STOP	0xB2	Parar chave de mídia
VK_MEDIA_PLAY_PAUSE	0xB3	Tecla Reproduzir/Pausar Mídia
VK_LAUNCH_MAIL	0xB4	Chave iniciar email
VK_LAUNCH_MEDIA_SELECT	0xB5	Selecione Chave de mídia
VK_LAUNCH_APP1	0xB6	Iniciar chave do Aplicativo 1
VK_LAUNCH_APP2	0xB7	Iniciar chave do Aplicativo 2
-	0xB8-B9	Reservado
VK_OEM_1	0xBA	Usado para caracteres diversos; pode variar de acordo com o teclado. Para o teclado padrão dos EUA, a tecla ';:'
VK_OEM_PLUS	0xBB	Para qualquer país/região, a chave '+'
VK_OEM_COMMA	0xBC	Para qualquer país/região, a chave ','
VK_OEM_MINUS	0xBD	Para qualquer país/região, a chave '-'

CONSTANTE	VALOR	DESCRIÇÃO
`VK_OEM_PERIOD`	0xBE	Para qualquer país/região, a chave '.'
`VK_OEM_2`	0xBF	Usado para caracteres diversos; pode variar de acordo com o teclado. Para o teclado padrão dos EUA, o '/?' chave
`VK_OEM_3`	0xC0	Usado para caracteres diversos; pode variar de acordo com o teclado. Para o teclado padrão dos EUA, a tecla ' ` ~'
`-`	0xC1-D7	Reservado
`-`	0xD8-DA	Não atribuído
`VK_OEM_4`	0xDB	Usado para caracteres diversos; pode variar de acordo com o teclado. Para o teclado padrão dos EUA, a tecla ' [{'
`VK_OEM_5`	0xDC	Usado para caracteres diversos; pode variar de acordo com o teclado. Para o teclado padrão dos EUA, a \ | tecla ' '
`VK_OEM_6`	0xdd	Usado para caracteres diversos; pode variar de acordo com o teclado. Para o teclado padrão dos EUA, a tecla '] }'
`VK_OEM_7`	0xDE	Usado para caracteres diversos; pode variar de acordo com o teclado. Para o teclado padrão dos EUA, a tecla 'aspas simples/aspas duplas'
`VK_OEM_8`	0xdf	Usado para caracteres diversos; pode variar de acordo com o teclado.
`-`	0xE0	Reservado
	0xE1	Específico do OEM
`VK_OEM_102`	0xE2	As teclas no teclado padrão dos EUA ou a tecla no teclado que não é us `<>` `\\|` 102-key
	0xE3-E4	Específico do OEM
`VK_PROCESSKEY`	0xE5	Chave de PROCESSO do IME
	0xE6	Específico do OEM

CONSTANTE	VALOR	DESCRIÇÃO
`VK_PACKET`	0xE7	Usada para passar a caracteres Unicode como se fossem pressionamentos de teclas. A chave é a palavra baixa de um valor de Chave Virtual de `VK_PACKET` 32 bits usado para métodos de entrada não teclado. Para obter mais informações, consulte Comentário `KEYBDINPUT` em `SendInput` , , `WM_KEYDOWN` e `WM_KEYUP`
`-`	0xE8	Não atribuído
	0xE9-F5	Específico do OEM
`VK_ATTN`	0xF6	Tecla Attn
`VK_CRSEL`	0xF7	Chave CrSel
`VK_EXSEL`	0xF8	Chave ExSel
`VK_EREOF`	0xF9	Apagar chave EOF
`VK_PLAY`	0xFA	Tecla de reprodução
`VK_ZOOM`	0xFB	Tecla de zoom
`VK_NONAME`	0xfc	Reservado
`VK_PA1`	0xFD	Tecla PA1
`VK_OEM_CLEAR`	0xFE	Limpar chave

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h

Entrada do mouse (entrada do teclado e mouse)

15/04/2022 • 11 minutes to read

Esta seção descreve como o sistema fornece entrada do mouse para seu aplicativo e como o aplicativo recebe e processa essa entrada.

Nesta seção

TÓPICO	DESCRIÇÃO
Sobre a entrada do mouse	Este tópico discute a entrada do mouse.
Usando a entrada do mouse	Esta seção aborda as tarefas associadas à entrada do mouse.
Referência de entrada do mouse	

Funções

NOME	DESCRIÇÃO
_Trackmouseevent	Poste mensagens quando o ponteiro do mouse sair de uma janela ou passar o mouse sobre uma janela por um período especificado. Essa função **chamará TrackMouseEvent** se ele existir, caso contrário, ele o emula.
DragDetect	Captura o mouse e acompanha seu movimento até que o usuário libere o botão esquerdo, pressione a tecla ESC ou mova o mouse para fora do retângulo de arrastar ao redor do ponto especificado. A largura e a altura do retângulo de arrastar são especificadas pelos valores **SM _ CXDRAG** e **SM _ CYDRAG** retornados pela **função GetSystemMetrics.**
EnableMouseInPointer	Permite que o mouse atue como um dispositivo apontador.
GetCapture	Recupera um alça para a janela (se há) que capturou o mouse. Somente uma janela por vez pode capturar o mouse; essa janela recebe a entrada do mouse se o cursor está ou não dentro de suas bordas.
GetDoubleClickTime	Recupera a hora atual de clique duplo para o mouse. Um clique duplo é uma série de dois cliques do botão do mouse, o segundo ocorrendo dentro de um horário especificado após o primeiro. O tempo de clique duplo é o número máximo de milissegundos que podem ocorrer entre o primeiro e o segundo clique de um clique duplo.
GetMouseMovePointsEx	Recupera um histórico de até 64 coordenadas anteriores do mouse ou caneta.

NOME	DESCRIÇÃO
Releasecapture	Libera a captura do mouse de uma janela no thread atual e restaura o processamento normal de entrada do mouse. Uma janela que capturou o mouse recebe toda a entrada do mouse, independentemente da posição do cursor, exceto quando um botão do mouse é clicado enquanto o cursor está na janela de outro thread.
Setcapture	Define a captura do mouse para a janela especificada que pertence ao thread atual. SetCapture captura a entrada do mouse quando o mouse está sobre a janela de captura ou quando o botão do mouse foi pressionado enquanto o mouse estava sobre a janela de captura e o botão ainda está inoportuno. Somente uma janela por vez pode capturar o mouse. Se o cursor do mouse estiver sobre uma janela criada por outro thread, o sistema direcionará a entrada do mouse para a janela especificada somente se um botão do mouse estiver inoossado.
SetDoubleClickTime	Define a hora de clique duplo para o mouse. Um clique duplo é uma série de dois cliques de um botão do mouse, o segundo ocorrendo dentro de um horário especificado após o primeiro. O tempo de clique duplo é o número máximo de milissegundos que podem ocorrer entre o primeiro e o segundo cliques de um clique duplo.
SwapMouseButton	Inverte ou restaura o significado dos botões esquerdo e direito do mouse.
Trackmouseevent	Poste mensagens quando o ponteiro do mouse sair de uma janela ou passar o mouse sobre uma janela por um período especificado.

A função a seguir está obsoleta.

FUNÇÃO	DESCRIÇÃO
evento _ mouse	Sintetiza os cliques de botão e movimento do mouse.

Notificações

NOME	DESCRIÇÃO
WM _ CAPTURECHANGED	Enviado para a janela que está perdendo a captura do mouse.
WM _ LBUTTONDBLCLK	Postado quando o usuário clica duas vezes no botão esquerdo do mouse enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.

NOME	DESCRIÇÃO
WM _ LBUTTONDOWN	Postado quando o usuário pressiona o botão esquerdo do mouse enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.
WM _ LBUTTONUP	Postado quando o usuário libera o botão esquerdo do mouse enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.
WM _ MBUTTONDBLCLK	Postado quando o usuário clica duas vezes no botão do meio do mouse enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.
WM _ MBUTTONDOWN	Postado quando o usuário pressiona o botão do meio do mouse enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.
WM _ MBUTTONUP	Postado quando o usuário libera o botão do meio do mouse enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.
WM _ MOUSEACTIVATE	Enviado quando o cursor está em uma janela inativa e o usuário pressiona um botão do mouse. A janela pai receberá essa mensagem somente se a janela filho passá-la para a função DefWindowProc.
WM _ MOUSEHOVER	Postado em uma janela quando o cursor passar o mouse sobre a área do cliente da janela pelo período de tempo especificado em uma chamada anterior para TrackMouseEvent.
WM _ MOUSEHWHEEL	Enviado para a janela de foco quando a roda de rolagem horizontal do mouse é inclinada ou girada. A função DefWindowProcpropaga a mensagem para o pai da janela. Não deve haver nenhum encaminhamento interno da mensagem, pois **DefWindowProc** a propaga pela cadeia pai até encontrar uma janela que a processe.
WM _ MOUSELEAVE	Postado em uma janela quando o cursor sai da área do cliente da janela especificada em uma chamada anterior para TrackMouseEvent.
WM _ MOUSEMOVE	Postado em uma janela quando o cursor se move. Se o mouse não for capturado, a mensagem será postada na janela que contém o cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.

NOME	DESCRIÇÃO
WM _ MOUSEWHEEL	Enviado para a janela de foco quando a roda do mouse é girada. A **função DefWindowProc** propaga a mensagem para o pai da janela. Não deve haver nenhum encaminhamento interno da mensagem, pois **DefWindowProc** a propaga pela cadeia pai até encontrar uma janela que a processe.
WM _ NCHITTEST	Enviado para uma janela para determinar qual parte da janela corresponde a uma coordenada de tela específica. Isso pode acontecer, por exemplo, quando o cursor se move, quando um botão do mouse é pressionado ou liberado ou em resposta a uma chamada para uma função como **WindowFromPoint**. Se o mouse não for capturado, a mensagem será enviada para a janela abaixo do cursor. Caso contrário, a mensagem será enviada para a janela que capturou o mouse.
WM _ NCLBUTTONDBLCLK	Postado quando o usuário clica duas vezes no botão esquerdo do mouse enquanto o cursor está dentro da área não dependente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.
WM _ NCLBUTTONDOWN	Postado quando o usuário pressiona o botão esquerdo do mouse enquanto o cursor está dentro da área não dependente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.
WM _ NCLBUTTONUP	Postado quando o usuário libera o botão esquerdo do mouse enquanto o cursor está dentro da área não dependente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.
WM _ NCMBUTTONDBLCLK	Postado quando o usuário clica duas vezes no botão do meio do mouse enquanto o cursor está dentro da área não dependente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.
WM _ NCMBUTTONDOWN	Postado quando o usuário pressiona o botão do meio do mouse enquanto o cursor está dentro da área não dependente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.
WM _ NCMBUTTONUP	Postado quando o usuário libera o botão do meio do mouse enquanto o cursor está dentro da área não dependente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.
WM _ NCMOUSEHOVER	Postado em uma janela quando o cursor passar o mouse sobre a área não dependente da janela pelo período de tempo especificado em uma chamada anterior para **TrackMouseEvent**.

NOME	DESCRIÇÃO
WM _ NCMOUSELEAVE	Postado em uma janela quando o cursor deixa a área não dependente da janela especificada em uma chamada anterior para TrackMouseEvent.
WM _ NCMOUSEMOVE	Postado em uma janela quando o cursor é movido dentro da área não dependente da janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.
WM _ NCRBUTTONDBLCLK	Postado quando o usuário clica duas vezes no botão direito do mouse enquanto o cursor está dentro da área não dependente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.
WM _ NCRBUTTONDOWN	Postado quando o usuário pressiona o botão direito do mouse enquanto o cursor está dentro da área não dependente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.
WM _ NCRBUTTONUP	Postado quando o usuário libera o botão direito do mouse enquanto o cursor está dentro da área não dependente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.
WM _ NCXBUTTONDBLCLK	Postado quando o usuário clica duas vezes no primeiro ou segundo botão X enquanto o cursor está na área não dependente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.
WM _ NCXBUTTONDOWN	Postado quando o usuário pressiona o primeiro ou segundo botão X enquanto o cursor está na área não dependente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.
WM _ NCXBUTTONUP	Postado quando o usuário libera o primeiro ou segundo botão X enquanto o cursor está na área não dependente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.
WM _ RBUTTONDBLCLK	Postado quando o usuário clica duas vezes no botão direito do mouse enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.
WM _ RBUTTONDOWN	Postado quando o usuário pressiona o botão direito do mouse enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.

NOME	DESCRIÇÃO
WM _ RBUTTONUP	Postado quando o usuário libera o botão direito do mouse enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.
WM _ XBUTTONDBLCLK	Postado quando o usuário clica duas vezes no primeiro ou segundo botão X enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.
WM _ XBUTTONDOWN	Postado quando o usuário pressiona o primeiro ou segundo botão X enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.
WM _ XBUTTONUP	Postado quando o usuário libera o primeiro ou segundo botão X enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.

Estruturas

NOME	DESCRIÇÃO
MOUSEMOVEPOINT	Contém informações sobre o local do mouse em coordenadas de tela.
TRACKMOUSEEVENT	Usado pela **função TrackMouseEvent** para acompanhar quando o ponteiro do mouse sai de uma janela ou passar o mouse sobre uma janela por um período especificado.

Sobre a entrada do mouse

15/04/2022 • 21 minutes to read

O mouse é um dispositivo importante, mas opcional, de entrada do usuário para aplicativos. Um aplicativo bem escrito deve incluir uma interface do mouse, mas não deve depender exclusivamente do mouse para adquirir a entrada do usuário. O aplicativo também deve fornecer suporte completo ao teclado.

Um aplicativo recebe a entrada do mouse na forma de mensagens enviadas ou postadas em suas janelas.

Esta seção contém os seguintes tópicos:

- Mouse Cursor
- Captura do mouse
- Mouse ClickLock
- Configuração do mouse
- XBUTTONs
- Mensagens do mouse
  - Mensagens do mouse da área do cliente
  - Mensagens do mouse de área não delimitada
  - A mensagem WM _ NCHITTEST
- Mouse Sonar
- Ressusução do mouse
- A roda do mouse
- Ativação de janela

Mouse Cursor

Quando o usuário move o mouse, o sistema move um bitmap na tela chamado *cursor do mouse*. O cursor do mouse contém um ponto de pixel único chamado ponto de calor , um ponto que o sistema rastreia e reconhece como a posição do cursor. Quando ocorre um evento do mouse, a janela que contém o ponto quente normalmente recebe a mensagem do mouse resultante do evento. A janela não precisa estar ativa ou ter o foco do teclado para receber uma mensagem do mouse.

O sistema mantém uma variável que controla a velocidade do mouse, ou seja, a distância que o cursor se move quando o usuário move o mouse. Você pode usar a **função SystemParametersInfo** com o **sinalizador SPI _ GETMOUSE** ou **SPI _ SETMOUSE** para recuperar ou definir a velocidade do mouse. Para obter mais informações sobre cursores do mouse, consulte Cursores.

Captura do mouse

O sistema normalmente posta uma mensagem do mouse na janela que contém o ponto quente do cursor quando ocorre um evento do mouse. Um aplicativo pode alterar esse comportamento usando a **função SetCapture** para rotear mensagens do mouse para uma janela específica. A janela recebe todas as mensagens do mouse até que o aplicativo chama a função **ReleaseCapture** ou especifica outra janela de captura ou até que o usuário clique em uma janela criada por outro thread.

Quando a captura do mouse muda, o sistema envia uma **mensagem WM _ CAPTURECHANGED** para a janela que está perdendo a captura do mouse. O *parâmetro lParam* da mensagem especifica um ponteiro para a janela que está ganhando a captura do mouse.

Somente a janela de primeiro plano pode capturar a entrada do mouse. Quando uma janela em segundo plano tenta capturar a entrada do mouse, ela recebe mensagens somente para eventos do mouse que ocorrem quando o ponto quente do cursor está dentro da parte visível da janela.

Capturar a entrada do mouse será útil se uma janela deve receber toda a entrada do mouse, mesmo quando o cursor se move para fora da janela. Por exemplo, um aplicativo normalmente rastreia a posição do cursor após um evento de botão do mouse para baixo, seguindo o cursor até que ocorra um evento de aumento do botão do mouse. Se um aplicativo não tiver capturado a entrada do mouse e o usuário liberar o botão do mouse fora da janela, a janela não receberá a mensagem de aplicação do botão.

Um thread pode usar a **função GetCapture** para determinar se uma de suas janelas capturou o mouse. Se uma das janelas do thread tiver capturado o mouse, **GetCapture** recuperará uma alça para a janela.

Mouse ClickLock

O recurso de acessibilidade ClickLock do mouse permite que um usuário bloqueie o botão principal do mouse após um único clique. Para um aplicativo, o botão ainda parece estar pressionado. Para desbloquear o botão, um aplicativo pode enviar qualquer mensagem do mouse ou o usuário pode clicar em qualquer botão do mouse. Esse recurso permite que um usuário faça combinações complexas de mouse de forma mais simples. Por exemplo, aqueles com determinadas limitações físicas podem realçar texto, arrastar objetos ou abrir menus com mais facilidade. Para obter mais informações, consulte os seguintes sinalizadores e os Comentários em **SystemParametersInfo**:

- **SPI _ GETMOUSECLICKLOCK**
- **SPI _ SETMOUSECLICKLOCK**
- **SPI _ GETMOUSECLICKLOCKTIME**
- **SPI _ SETMOUSECLICKLOCKTIME**

Configuração do mouse

Embora o mouse seja um dispositivo de entrada importante para aplicativos, nem todos os usuários necessariamente têm um mouse. Um aplicativo pode determinar se o sistema inclui um mouse passando o valor **_ MOUSEPRESENT do SM** para a **função GetSystemMetrics.**

Windows dá suporte a um mouse com até três botões. Em um mouse de três botões, os botões são designados como os botões esquerdo, médio e direito. Mensagens e constantes nomeadas relacionadas aos botões do mouse usam as letras L, M e R para identificar os botões. O botão em um mouse de botão único é considerado o botão esquerdo. Embora Windows suporte a um mouse com vários botões, a maioria dos aplicativos usa o botão esquerdo principalmente e os outros minimamente, se for o caso.

Os aplicativos também podem dar suporte a uma roda do mouse. A roda do mouse pode ser pressionada ou girada. Quando a roda do mouse é pressionada, ela atua como o botão central (terceiro), enviando mensagens normais de botão central para seu aplicativo. Quando ela é girada, uma mensagem de roda é enviada para seu aplicativo. Para obter mais informações, consulte a seção Roda do Mouse.

Os aplicativos podem dar suporte a botões application-command. Esses botões, chamados botões X, foram projetados para permitir o acesso mais fácil a um navegador da Internet, email eletrônico e serviços de mídia. Quando um botão X é pressionado, uma **mensagem _ WM APPCOMMAND** é enviada ao seu aplicativo. Para obter mais informações, consulte a descrição na mensagem **_ WM APPCOMMAND.**

Um aplicativo pode determinar o número de botões no mouse passando o valor **_ de SM CMOUSEBUTTONS** para a **função GetSystemMetrics.** Para configurar o mouse para um usuário à esquerda, o aplicativo pode usar a função **SwapMouseButton** para inverter o significado dos botões esquerdo e direito do mouse. Passar o **valor _ SPI SETMOUSEBUTTONSWAP** para a função **SystemParametersInfo** é outra maneira de reverter o significado dos botões. Observe, no entanto, que o mouse é um recurso compartilhado, portanto, reverter o

significado dos botões afeta todos os aplicativos.

XBUTTONs

Windows dá suporte a um mouse com cinco botões. Além dos botões esquerdo, médio e direito, há XBUTTON1 e XBUTTON2, que fornecem navegação para trás e para frente ao usar o navegador.

O gerenciador de janelas dá suporte a XBUTTON1 e XBUTTON2 por meio das mensagens **WM _ XBUTTON * _ *e* _ WM _ NCXBUTTON * _ . *A HIWORD do* _ WPARAM** nessas mensagens contém um sinalizador que indica qual botão X foi pressionado. Como essas mensagens do mouse também se ajustam entre as constantes **WM _ MOUSEFIRST** e **WM _ MOUSELAST,** um aplicativo pode filtrar todas as mensagens do mouse com **GetMessage** ou **PeekMessage**.

Os seguintes suportes são XBUTTON1 e XBUTTON2:

- **WM _ APPCOMMAND**
- **WM _ NCXBUTTONDBLCLK**
- **WM _ NCXBUTTONDOWN**
- **WM _ NCXBUTTONUP**
- **WM _ XBUTTONDBLCLK**
- **WM _ XBUTTONDOWN**
- **WM _ XBUTTONUP**
- **MOUSEHOOKSTRUCTEX**

As SEGUINTES APIs foram modificadas para dar suporte a estes botões:

- **evento _ mouse**
- **ShellProc**
- **MSLLHOOKSTRUCT**
- **MOUSEINPUT**
- **WM _ PARENTNOTIFY**

É improvável que uma janela filho em um aplicativo de componente seja capaz de implementar comandos diretamente para XBUTTON1 e XBUTTON2. Portanto, **DefWindowProc** envia uma **mensagem WM _ APPCOMMAND** para uma janela quando um botão X é clicado. **DefWindowProc** também envia a mensagem **WM _ APPCOMMAND** para sua janela pai. Isso é semelhante à maneira como os menus de contexto são invocados com um clique com o botão direito do mouse–**DefWindowProc** envia uma mensagem **WM _ CONTEXTMENU** para o menu e também a envia para seu pai. Além disso, se **DefWindowProc** receber uma mensagem **WM _ APPCOMMAND** para uma janela de nível superior, ele chamará um gancho de shell com o código HSHELL _ APPCOMMAND.

Há suporte para os teclados que têm chaves extras para funções de navegador, funções de mídia, lançamento de aplicativos e gerenciamento de energia. Para obter mais informações, consulte Teclas de teclado para navegação e outras funções.

Mensagens do mouse

O mouse gera um evento de entrada quando o usuário move o mouse ou pressiona ou libera um botão do mouse. O sistema converte eventos de entrada do mouse em mensagens e os posta na fila de mensagens do thread apropriado. Quando as mensagens do mouse são postadas mais rapidamente do que um thread pode processá-las, o sistema descarta tudo, menos a mensagem mais recente do mouse.

Uma janela recebe uma mensagem do mouse quando ocorre um evento do mouse enquanto o cursor está dentro das bordas da janela ou quando a janela captura o mouse. As mensagens do mouse são divididas em

dois grupos: mensagens de área do cliente e mensagens de área não cliente. Normalmente, um aplicativo processa mensagens de área do cliente e ignora mensagens de área não cliente.

Esta seção contém os seguintes tópicos:

- Mensagens do mouse da área do cliente
- Mensagens do mouse de área não delimitada
- A mensagem WM _ NCHITTEST

Mensagens do mouse da área do cliente

Uma janela recebe uma mensagem do mouse da área do cliente quando ocorre um evento do mouse na área do cliente da janela. O sistema posta a **mensagem WM _ MOUSEMOVE** na janela quando o usuário move o cursor dentro da área do cliente. Ele posta uma das mensagens a seguir quando o usuário pressiona ou libera um botão do mouse enquanto o cursor está dentro da área do cliente.

MENSAGEM	SIGNIFICADO
WM _ LBUTTONDBLCLK	O botão esquerdo do mouse foi clicado duas vezes.
WM _ LBUTTONDOWN	O botão esquerdo do mouse foi pressionado.
WM _ LBUTTONUP	O botão esquerdo do mouse foi liberado.
WM _ MBUTTONDBLCLK	O botão do meio do mouse foi clicado duas vezes.
WM _ MBUTTONDOWN	O botão do meio do mouse foi pressionado.
WM _ MBUTTONUP	O botão do meio do mouse foi liberado.
WM _ RBUTTONDBLCLK	O botão direito do mouse foi clicado duas vezes.
WM _ RBUTTONDOWN	O botão direito do mouse foi pressionado.
WM _ RBUTTONUP	O botão direito do mouse foi liberado.
WM _ XBUTTONDBLCLK	Um botão X do mouse foi clicado duas vezes.
WM _ XBUTTONDOWN	Um botão X do mouse foi pressionado.
WM _ XBUTTONUP	Um botão X do mouse foi liberado.

Além disso, um aplicativo pode chamar a **função TrackMouseEvent** para que o sistema envie duas outras mensagens. Ele posta a **mensagem WM _ MOUSEHOVER** quando o cursor passar o mouse sobre a área do cliente por um determinado período de tempo. Ele posta a **mensagem WM _ MOUSELEAVE** quando o cursor sai da área do cliente.

Parâmetros de mensagem

O *parâmetro lParam* de uma mensagem do mouse da área do cliente indica a posição do ponto de acionamento do cursor. A palavra de ordem baixa indica a coordenada X do ponto de calor e a palavra de ordem alta indica a coordenada y. As coordenadas são especificadas nas coordenadas do cliente. No sistema de coordenadas do cliente, todos os pontos na tela são especificados em relação às coordenadas (0,0) do canto superior esquerdo da área do cliente.

O *parâmetro wParam* contém sinalizadores que indicam o status dos outros botões do mouse e as teclas CTRL e

SHIFT no momento do evento do mouse. Você pode verificar esses sinalizadores quando o processamento da mensagem do mouse depende do estado de outro botão do mouse ou da tecla CTRL ou SHIFT. O *parâmetro wParam* pode ser uma combinação dos valores a seguir.

VALOR	DESCRIÇÃO
CONTROLE _ MK	A tecla CTRL está inocizada.
MK _ LBUTTON	O botão esquerdo do mouse está ino mouse.
MK _ MBUTTON	O botão do meio do mouse está ino mouse.
MK _ RBUTTON	O botão direito do mouse está ino mouse.
SHIFT _ da MK	A tecla SHIFT está inobada.
MK _ XBUTTON1	O primeiro botão X está ino mouse.
MK _ XBUTTON2	O segundo botão X está ino mouse.

Double-Click mensagens

O sistema gera uma mensagem de clique duplo quando o usuário clica no botão do mouse duas vezes em sucessão rápida. Quando o usuário clica em um botão, o sistema estabelece um retângulo centralizado em torno do ponto de contato do cursor. Ele também marca a hora em que o clique ocorreu. Quando o usuário clica no mesmo botão uma segunda vez, o sistema determina se o ponto de calor ainda está dentro do retângulo e calcula o tempo decorrido desde o primeiro clique. Se o ponto de calor ainda estiver dentro do retângulo e o tempo decorrido não exceder o valor de tempo limite de clique duplo, o sistema gerará uma mensagem de clique duplo.

Um aplicativo pode obter e definir valores de tempo-out de clique duplo usando as funções **GetDoubleClickTime** e **SetDoubleClickTime,** respectivamente. Como alternativa, o aplicativo pode definir o valor de tempo-de-tempo-de-clique duplo usando o **sinalizador SPI _ SETDOUBLECLICKTIME** com a **função SystemParametersInfo.** Ele também pode definir o tamanho do retângulo que o sistema usa para detectar cliques duplos passando os sinalizadores **SPI _ SETDOUBLECLKWIDTH** e **SPI _ SETDOUBLECLKHEIGHT** para **SystemParametersInfo**. Observe, no entanto, que definir o valor de tempo-de-tempo-de-clique duplo e o retângulo afeta todos os aplicativos.

Uma janela definida pelo aplicativo não recebe, por padrão, mensagens de clique duplo. Devido à sobrecarga do sistema envolvida na geração de mensagens de clique duplo, essas mensagens são geradas somente para janelas que pertencem a classes que têm o estilo de classe **_ DBLCLKS CS.** Seu aplicativo deve definir esse estilo ao registrar a classe de janela. Para obter mais informações, consulte Classes de janela.

Uma mensagem de clique duplo é sempre a terceira mensagem em uma série de quatro mensagens. As duas primeiras mensagens são as mensagens de botão para baixo e de botão para cima geradas pelo primeiro clique. O segundo clique gera a mensagem de clique duplo seguida por outra mensagem de botão. Por exemplo, clicar duas vezes com o botão esquerdo do mouse gera a seguinte sequência de mensagens:

1. **LBUTTONDOWN do WM _**
2. **LBUTTONUP do WM _**
3. **LBUTTONDBLCLK do WM _**
4. **LBUTTONUP do WM _**

Como uma janela sempre recebe uma mensagem de botão para baixo antes de receber uma mensagem de clique duplo, um aplicativo geralmente usa uma mensagem de clique duplo para estender uma tarefa iniciada

durante uma mensagem de botão para baixo. por exemplo, quando o usuário clica em uma cor na paleta de cores de Microsoft Paint, Paint exibe a cor selecionada ao lado da paleta. quando o usuário clica duas vezes em uma cor, Paint exibe a cor e abre a caixa de diálogo **editar cores** .

Mensagens de mouse de área não cliente

Uma janela recebe uma mensagem de mouse de área não cliente quando ocorre um evento do mouse em qualquer parte de uma janela, exceto a área do cliente. A área não cliente de uma janela consiste em sua borda, barra de menus, barra de título, barra de rolagem, menu janela, botão minimizar e botão Maximizar.

O sistema gera mensagens de área não cliente principalmente para seu próprio uso. Por exemplo, o sistema usa mensagens de área não cliente para alterar o cursor para uma seta de duas pontas quando o ponto de acesso do cursor se move para a borda de uma janela. Uma janela deve passar mensagens de mouse de área não cliente para a função **DefWindowProc** para aproveitar a interface do mouse interna.

Há uma mensagem de mouse de área não cliente correspondente para cada mensagem de mouse de área do cliente. Os nomes dessas mensagens são semelhantes, exceto pelo fato de que as constantes nomeadas para as mensagens de área não cliente incluem o NC de letras. Por exemplo, mover o cursor na área não cliente gera uma mensagem do **WM _ NCMOUSEMOVE** e pressionar o botão esquerdo do mouse enquanto o cursor está na área não cliente gera uma mensagem do **WM _ NCLBUTTONDOWN** .

O parâmetro *lParam* de uma mensagem de mouse de área não cliente é uma estrutura que contém as coordenadas x e y do ponto de acesso do cursor. Ao contrário das coordenadas das mensagens de mouse da área do cliente, as coordenadas são especificadas em coordenadas da tela, em vez de coordenadas do cliente. No sistema de coordenadas de tela, todos os pontos na tela são relativos às coordenadas (0, 0) do canto superior esquerdo da tela.

O parâmetro *wParam* contém um valor de teste de clique, um valor que indica em que local na área não cliente ocorreu o evento de mouse. A seção a seguir explica a finalidade dos valores de teste de clique.

A mensagem do WM _ NCHITTEST

Sempre que um evento de mouse ocorre, o sistema envia uma mensagem do **WM _ NCHITTEST** para a janela que contém o ponto de acesso do cursor ou a janela que capturou o mouse. O sistema usa essa mensagem para determinar se deve ser enviada uma mensagem de cliente ou de mouse de área de clientes. Um aplicativo que deve receber o movimento do mouse e as mensagens do botão do mouse devem passar a mensagem do **WM _ NCHITTEST** para a função **DefWindowProc** .

O parâmetro *lParam* da mensagem **de _ NCHITTEST do WM** contém as coordenadas de tela do ponto de acesso do cursor. A função **DefWindowProc** examina as coordenadas e retorna um valor de teste de clique que indica o local do ponto de acesso. O valor de teste de clique pode ser um dos valores a seguir.

VALOR	LOCAL DO PONTO DE ACESSO
HTBORDER	Na borda de uma janela que não tem uma borda de dimensionamento.
HTBOTTOM	Na borda inferior horizontal de uma janela.
HTBOTTOMLEFT	No canto inferior esquerdo de uma borda de janela.
HTBOTTOMRIGHT	No canto inferior direito de uma borda de janela.
HTCAPTION	Em uma barra de título.
HTCLIENT	Em uma área de cliente.

VALOR	LOCAL DO PONTO DE ACESSO
HTCLOSE	Em um botão **fechar** .
HTERROR	Na tela de fundo ou em uma linha divisória entre janelas (o mesmo que HTNOWHERE, exceto que a função DefWindowProc produz um aviso sonoro do sistema para indicar um erro).
HTGROWBOX	Em uma caixa de tamanho (igual a **HTSIZE**).
HTHELP	Em um botão de **ajuda** .
HTHSCROLL	Em uma barra de rolagem horizontal.
HTLEFT	Na borda esquerda de uma janela.
HTMENU	Em um menu.
HTMAXBUTTON	Em um botão **maximizar** .
HTMINBUTTON	Em um botão **minimizar** .
HTNOWHERE	Na tela de fundo ou em uma linha divisória entre janelas.
HTREDUCE	Em um botão **minimizar** .
HTRIGHT	Na borda direita de uma janela.
HTSIZE	Em uma caixa de tamanho (igual a **HTGROWBOX**).
HTSYSMENU	Em um menu do **sistema** ou em um botão **fechar** em uma janela filho.
HTTOP	Na borda superior horizontal de uma janela.
HTTOPLEFT	No canto superior esquerdo de uma borda de janela.
HTTOPRIGHT	No canto superior direito de uma borda de janela.
HTTRANSPARENT	Em uma janela atualmente coberta por outra janela no mesmo thread.
HTVSCROLL	Na barra de rolagem vertical.
HTZOOM	Em um botão **maximizar** .

Se o cursor estiver na área de cliente de uma janela, DefWindowProc retornará o valor de teste de clique **HTCLIENT** para o procedimento de janela. Quando o procedimento de janela retorna esse código para o sistema, o sistema converte as coordenadas de tela do ponto de acesso do cursor para as coordenadas do cliente e, em seguida, posta a mensagem de mouse da área do cliente apropriada.

A função DefWindowProc retorna um dos outros valores de teste de clique quando o ponto de acesso do cursor está em uma área não cliente da janela. Quando o procedimento de janela retorna um desses valores de

teste de clique, o sistema posta uma mensagem de mouse de área não cliente, colocando o valor de teste de clique no parâmetro *wParam* da mensagem e as coordenadas do cursor no parâmetro *lParam* .

Mouse de sonar

O recurso de acessibilidade do sonar do mouse mostra brevemente vários círculos concêntricos em volta do ponteiro quando o usuário pressiona e libera a tecla CTRL. Esse recurso ajuda um usuário a localizar o ponteiro do mouse em uma tela que está obstruída ou com resolução definida como alta, em um monitor de baixa qualidade ou para usuários com visão inemparelhada. Para obter mais informações, consulte os seguintes sinalizadores em SystemParametersInfo:

SPI _ GETMOUSESONAR

SPI _ SETMOUSESONAR

Desaparecer do mouse

O recurso de acessibilidade do mouse desaparecer oculta o ponteiro quando o usuário está digitando. O ponteiro do mouse reaparece quando o usuário move o mouse. Esse recurso mantém o ponteiro de obscurecer o texto que está sendo digitado, por exemplo, em um email ou em outro documento. Para obter mais informações, consulte os seguintes sinalizadores em SystemParametersInfo:

SPI _ GETMOUSEVANISH

SPI _ SETMOUSEVANISH

A roda do mouse

A roda do mouse combina os recursos de uma roda e um botão do mouse. A roda tem entalhes discretos e espaçados uniformemente. Quando você gira a roda, uma mensagem de roda é enviada ao seu aplicativo à medida que cada entalhe é encontrado. o botão de roda também pode operar como um botão normal Windows meio (terceiro). Pressionar e liberar a roda do mouse envia mensagens padrão do WM _ MBUTTONUP e do WM _ MBUTTONDOWN . Clicar duas vezes no terceiro botão envia a mensagem padrão do WM _ MBUTTONDBLCLK .

A roda do mouse tem suporte por meio da mensagem do WM _ MOUSEWHEEL .

Girar o mouse envia a mensagem do WM _ MOUSEWHEEL para a janela de foco. A função DefWindowProc propaga a mensagem para o pai da janela. Não deve haver nenhum encaminhamento interno da mensagem, pois **DefWindowProc** propaga a cadeia pai até que uma janela que o processa seja encontrada.

Determinando o número de linhas de rolagem

Os aplicativos devem usar a função SystemParametersInfo para recuperar o número de linhas que um documento rola para cada operação de rolagem (entalhe de roda). Para recuperar o número de linhas, um aplicativo faz a seguinte chamada:

```
SystemParametersInfo(SPI_GETWHEELSCROLLLINES, 0, pulScrollLines, 0)
```

A variável "pulScrollLines" aponta para um valor inteiro sem sinal que recebe o número sugerido de linhas para rolar quando a roda do mouse é girada sem Chaves modificadoras:

- Se esse número for 0, não deverá ocorrer nenhuma rolagem.
- Se esse número for o **Wheel _ PAGESCROLL**, um volante deverá ser interpretado como um clique uma vez nas regiões Page Down ou Page up da barra de rolagem.
- Se o número de linhas a rolar for maior que o número de linhas visíveis, a operação de rolagem também

deverá ser interpretada como uma operação Page Down ou Page up.

O valor padrão para o número de linhas de rolagem será 3. Se um usuário alterar o número de linhas de rolagem usando a folha de propriedades do mouse no painel de controle, o sistema operacional transmitirá uma mensagem do **WM _ SETTINGCHANGE** para todas as janelas de nível superior com **SPI _ SETWHEELSCROLLLINES** especificado. Quando um aplicativo recebe a **mensagem _ SETTINGCHANGE do WM** , ele pode obter o novo número de linhas de rolagem chamando:

```
SystemParametersInfo(SPI_GETWHEELSCROLLLINES, 0, pulScrollLines, 0)
```

Controles que rolam

A tabela a seguir lista os controles com a funcionalidade de rolagem (incluindo linhas de rolagem definidas pelo usuário).

CONTROL	ROLAGEM
Controle de edição	Vertical e horizontal.
Controle de caixa de listagem	Vertical e horizontal.
Caixa de combinação	Quando não é descartada, cada rolagem recupera o próximo item ou o anterior. Quando desativadas, cada rolagem encaminha a mensagem para a caixa de listagem, que rola de forma adequada.
CMD (linha de comando)	Vertical.
Modo de exibição de árvore	Vertical e horizontal.
Exibição de Lista	Vertical e horizontal.
Rolagens para cima/para baixo	Um item por vez.
Rolagens do TrackBar	Um item por vez.
Edição avançada da Microsoft 1,0	Vertical. observe que o cliente Exchange tem suas próprias versões dos controles de exibição de lista e de exibição de árvore que não têm suporte à roda.
Edição avançada da Microsoft 2,0	Vertical.

Detectando um mouse com uma roda

Para determinar se um mouse com uma roda está conectado, chame **GetSystemMetrics** com **SM _ MOUSEWHEELPRESENT**. Um valor de retorno **true** indica que o mouse está conectado.

O exemplo a seguir é do procedimento de janela para um controle de edição de várias linhas:

```
BOOL ScrollLines(
     PWNDDATA pwndData,   //scrolls the window indicated
     int cLinesToScroll); //number of times

short gcWheelDelta; //wheel delta from roll
PWNDDATA pWndData; //pointer to structure containing info about the window
UINT gucWheelScrollLines=0;//number of lines to scroll on a wheel rotation

gucWheelScrollLines = SystemParametersInfo(SPI_GETWHEELSCROLLLINES,
                              0,
                              pulScrollLines,
                              0);

case WM_MOUSEWHEEL:
    /*
     * Do not handle zoom and datazoom.
     */
    if (wParam & (MK_SHIFT | MK_CONTROL)) {
        goto PassToDefaultWindowProc;
    }

    gcWheelDelta -= (short) HIWORD(wParam);
    if (abs(gcWheelDelta) >= WHEEL_DELTA && gucWheelScrollLines > 0)
    {
        int cLineScroll;

        /*
         * Limit a roll of one (1) WHEEL_DELTA to
         * scroll one (1) page.
         */
        cLineScroll = (int) min(
                (UINT) pWndData->ichLinesOnScreen - 1,
                gucWheelScrollLines);

        if (cLineScroll == 0) {
            cLineScroll++;
        }

        cLineScroll *= (gcWheelDelta / WHEEL_DELTA);
        assert(cLineScroll != 0);

        gcWheelDelta = gcWheelDelta % WHEEL_DELTA;
        return ScrollLines(pWndData, cLineScroll);
    }

    break;
```

Ativação de janela

Quando o usuário clica em uma janela de nível superior inativa ou na janela filho de uma janela de nível superior inativa, o sistema envia a mensagem do **WM _ MOUSEACTIVATE** (entre outras) para a janela de nível superior ou filho. O sistema envia essa mensagem depois de postar a mensagem **_ NCHITTEST do WM** na janela, mas antes de postar a mensagem de botão para baixo. Quando o **WM _ MOUSEACTIVATE** é passado para a função **DefWindowProc** , o sistema ativa a janela de nível superior e, em seguida, posta a mensagem de botão para baixo na janela de nível superior ou filho.

Ao processar o **WM _ MOUSEACTIVATE**, uma janela pode controlar se a janela de nível superior se torna a janela ativa como resultado de um clique do mouse e se a janela que foi clicada recebe a mensagem de botão inferior. Ele faz isso retornando um dos seguintes valores após o processamento do **WM _ MOUSEACTIVATE**.

VALOR	SIGNIFICADO
ativação do MA _	Ativa a janela e não descarta a mensagem do mouse.
MA _ NOativar	Não ativa a janela e não descarta a mensagem do mouse.
_ACTIVATEANDEAT ma	Ativa a janela e descarta a mensagem do mouse.
_NOACTIVATEANDEAT ma	Não ativa a janela, mas descarta a mensagem do mouse.

Confira também

Aproveitando o movimento do High-Definition mouse

Usando a entrada do mouse

15/04/2022 • 13 minutes to read

Esta seção aborda as tarefas associadas à entrada do mouse.

- Acompanhando o cursor do mouse
- Desenhando linhas com o mouse
- Processamento de uma mensagem de clique duplo
- Selecionando uma linha de texto
- Usando uma roda do mouse em um documento com objetos inseridos
- Recuperando o número de linhas de rolagem da roda do mouse

Acompanhando o cursor do mouse

Os aplicativos geralmente executam tarefas que envolvem o acompanhamento da posição do cursor do mouse. A maioria dos aplicativos de desenho, por exemplo, rastreia a posição do cursor do mouse durante operações de desenho, permitindo que o usuário desenhe na área do cliente de uma janela arrastando o mouse. Os aplicativos de processamento de palavras também acompanham o cursor, permitindo que o usuário selecione uma palavra ou bloco de texto clicando e arrastando o mouse.

O acompanhamento do cursor normalmente envolve o processamento das mensagens **WM _ LBUTTONDOWN, WM _ MOUSEMOVE** e **WM _ LBUTTONUP.** Uma janela determina quando começar a acompanhar o cursor verificando a posição do cursor fornecida no parâmetro *lParam* da mensagem **WM _ LBUTTONDOWN.** Por exemplo, um aplicativo de processamento de palavras começaria a acompanhar o cursor somente se a mensagem **WM _ LBUTTONDOWN** ocorresse enquanto o cursor estava em uma linha de texto, mas não se ele tivesse passado do final do documento.

Uma janela rastreia a posição do cursor processando o fluxo de mensagens **WM _ MOUSEMOVE** postadas na janela conforme o mouse se move. O processamento da **mensagem WM _ MOUSEMOVE** normalmente envolve uma operação repetitiva de pintura ou desenho na área do cliente. Por exemplo, um aplicativo de desenho pode redesenhar uma linha repetidamente à medida que o mouse se move. Uma janela usa a **mensagem WM _ LBUTTONUP** como um sinal para interromper o acompanhamento do cursor.

Além disso, um aplicativo pode chamar a **função TrackMouseEvent** para que o sistema envie outras mensagens úteis para acompanhar o cursor. O sistema posta a **mensagem WM _ MOUSEHOVER** quando o cursor passar o mouse sobre a área do cliente por um determinado período de tempo. Ele posta a **mensagem WM _ MOUSELEAVE** quando o cursor sai da área do cliente. As **mensagens WM _ NCMOUSEHOVER** e **WM _ NCMOUSELEAVE** são as mensagens correspondentes para as áreas não dependentes.

Desenhando linhas com o mouse

O exemplo nesta seção demonstra como acompanhar o cursor do mouse. Ele contém partes de um procedimento de janela que permite que o usuário desenhe linhas na área do cliente de uma janela arrastando o mouse.

Quando o procedimento de janela recebe uma mensagem **WM _ LBUTTONDOWN,** ele captura o mouse e salva as coordenadas do cursor, usando as coordenadas como o ponto inicial da linha. Ele também usa a **função ClipCursor** para limitar o cursor à área do cliente durante a operação de desenho de linha.

Durante a primeira **mensagem WM _ MOUSEMOVE,** o procedimento de janela desenha uma linha do ponto inicial para a posição atual do cursor. Durante as **mensagens WM _ MOUSEMOVE** subsequentes, o

procedimento de janela apaga a linha anterior desenhando sobre ela com uma cor de caneta invertida. Em seguida, ele desenha uma nova linha do ponto de partida para a nova posição do cursor.

A **mensagem WM _ LBUTTONUP** sinaliza o final da operação de desenho. O procedimento de janela libera a captura do mouse e libera o mouse da área do cliente.

```
LRESULT APIENTRY MainWndProc(HWND hwndMain, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    HDC hdc;                    // handle to device context
    RECT rcClient;              // client area rectangle
    POINT ptClientUL;           // client upper left corner
    POINT ptClientLR;           // client lower right corner
    static POINTS ptsBegin;     // beginning point
    static POINTS ptsEnd;       // new endpoint
    static POINTS ptsPrevEnd;   // previous endpoint
    static BOOL fPrevLine = FALSE; // previous line flag

    switch (uMsg)
    {
       case WM_LBUTTONDOWN:

            // Capture mouse input.

            SetCapture(hwndMain);

            // Retrieve the screen coordinates of the client area,
            // and convert them into client coordinates.

            GetClientRect(hwndMain, &rcClient);
            ptClientUL.x = rcClient.left;
            ptClientUL.y = rcClient.top;

            // Add one to the right and bottom sides, because the
            // coordinates retrieved by GetClientRect do not
            // include the far left and lowermost pixels.

            ptClientLR.x = rcClient.right + 1;
            ptClientLR.y = rcClient.bottom + 1;
            ClientToScreen(hwndMain, &ptClientUL);
            ClientToScreen(hwndMain, &ptClientLR);

            // Copy the client coordinates of the client area
            // to the rcClient structure. Confine the mouse cursor
            // to the client area by passing the rcClient structure
            // to the ClipCursor function.

            SetRect(&rcClient, ptClientUL.x, ptClientUL.y,
                ptClientLR.x, ptClientLR.y);
            ClipCursor(&rcClient);

            // Convert the cursor coordinates into a POINTS
            // structure, which defines the beginning point of the
            // line drawn during a WM_MOUSEMOVE message.

            ptsBegin = MAKEPOINTS(lParam);
            return 0;

        case WM_MOUSEMOVE:

            // When moving the mouse, the user must hold down
            // the left mouse button to draw lines.

            if (wParam & MK_LBUTTON)
            {

                // Retrieve a device context (DC) for the client area.
```

```
                hdc = GetDC(hwndMain);

                // The following function ensures that pixels of
                // the previously drawn line are set to white and
                // those of the new line are set to black.

                SetROP2(hdc, R2_NOTXORPEN);

                // If a line was drawn during an earlier WM_MOUSEMOVE
                // message, draw over it. This erases the line by
                // setting the color of its pixels to white.

                if (fPrevLine)
                {
                    MoveToEx(hdc, ptsBegin.x, ptsBegin.y,
                        (LPPOINT) NULL);
                    LineTo(hdc, ptsPrevEnd.x, ptsPrevEnd.y);
                }

                // Convert the current cursor coordinates to a
                // POINTS structure, and then draw a new line.

                ptsEnd = MAKEPOINTS(lParam);
                MoveToEx(hdc, ptsBegin.x, ptsBegin.y, (LPPOINT) NULL);
                LineTo(hdc, ptsEnd.x, ptsEnd.y);

                // Set the previous line flag, save the ending
                // point of the new line, and then release the DC.

                fPrevLine = TRUE;
                ptsPrevEnd = ptsEnd;
                ReleaseDC(hwndMain, hdc);
            }
            break;

        case WM_LBUTTONUP:

            // The user has finished drawing the line. Reset the
            // previous line flag, release the mouse cursor, and
            // release the mouse capture.

            fPrevLine = FALSE;
            ClipCursor(NULL);
            ReleaseCapture();
            return 0;

        case WM_DESTROY:
            PostQuitMessage(0);
            break;

        // Process other messages.
```

Processamento de uma mensagem de clique duplo

Para receber mensagens de clique duplo, uma janela deve pertencer a uma classe de janela que tenha o estilo de classe _ **DBLCLKS do CS.** Você definirá esse estilo ao registrar a classe de janela, conforme mostrado no exemplo a seguir.

```
BOOL InitApplication(HINSTANCE hInstance)
{
    WNDCLASS wc;

    wc.style = CS_DBLCLKS | CS_HREDRAW | CS_VREDRAW;
    wc.lpfnWndProc = (WNDPROC) MainWndProc;
    wc.cbClsExtra = 0;
    wc.cbWndExtra = 0;
    wc.hInstance = hInstance;
    wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);
    wc.hCursor = LoadCursor(NULL, IDC_IBEAM);
    wc.hbrBackground = GetStockObject(WHITE_BRUSH);
    wc.lpszMenuName = "MainMenu";
    wc.lpszClassName = "MainWClass";

    return RegisterClass(&wc);
}
```

Uma mensagem de clique duplo sempre é precedida por uma mensagem de botão para baixo. Por esse motivo, os aplicativos normalmente usam uma mensagem de clique duplo para estender uma tarefa que começou durante uma mensagem de botão para baixo.

Selecionando uma linha de texto

O exemplo nesta seção é retirado de um aplicativo de processamento de palavras simples. Ele inclui código que permite ao usuário definir a posição do aro clicando em qualquer lugar em uma linha de texto e selecionando (realça) uma linha de texto clicando duas vezes em qualquer lugar da linha.

```
LRESULT APIENTRY MainWndProc(HWND hwndMain, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    HDC hdc;                    // handle to device context
    TEXTMETRIC tm;              // font size data
    int i, j;                   // loop counters
    int cCR = 0;                // count of carriage returns
    char ch;                    // character from input buffer
    static int nBegLine;        // beginning of selected line
    static int nCurrentLine = 0; // currently selected line
    static int nLastLine = 0;   // last text line
    static int nCaretPosX = 0;  // x-coordinate of caret
    static int cch = 0;         // number of characters entered
    static int nCharWidth = 0;  // exact width of a character
    static char szHilite[128];  // text string to highlight
    static DWORD dwCharX;       // average width of characters
    static DWORD dwLineHeight;  // line height
    static POINTS ptsCursor;    // coordinates of mouse cursor
    static COLORREF crPrevText; // previous text color
    static COLORREF crPrevBk;   // previous background color
    static PTCHAR pchInputBuf;  // pointer to input buffer
    static BOOL fTextSelected = FALSE; // text-selection flag
    size_t * pcch;
    HRESULT hResult;

    switch (uMsg)
    {
        case WM_CREATE:

            // Get the metrics of the current font.

            hdc = GetDC(hwndMain);
            GetTextMetrics(hdc, &tm);
            ReleaseDC(hwndMain, hdc);

            // Save the average character width and height.
```

```
            dwCharX = tm.tmAveCharWidth;
            dwLineHeight = tm.tmHeight;

            // Allocate a buffer to store keyboard input.

            pchInputBuf = (LPSTR) GlobalAlloc(GPTR,
                BUFSIZE * sizeof(TCHAR));

            return 0;

        case WM_CHAR:
            switch (wParam)
            {
                case 0x08:  // backspace
                case 0x0A:  // linefeed
                case 0x1B:  // escape
                    MessageBeep( (UINT) -1);
                    return 0;

                case 0x09:  // tab

                    // Convert tabs to four consecutive spaces.

                    for (i = 0; i < 4; i++)
                        SendMessage(hwndMain, WM_CHAR, 0x20, 0);
                    return 0;

                case 0x0D:  // carriage return

                    // Record the carriage return, and position the
                    // caret at the beginning of the new line.

                    pchInputBuf[cch++] = 0x0D;
                    nCaretPosX = 0;
                    nCurrentLine += 1;
                    break;

                default:    / displayable character

                    ch = (char) wParam;
                    HideCaret(hwndMain);

                    // Retrieve the character's width, and display the
                    // character.

                    hdc = GetDC(hwndMain);
                    GetCharWidth32(hdc, (UINT) wParam, (UINT) wParam,
                        &nCharWidth);
                    TextOut(hdc, nCaretPosX,
                        nCurrentLine * dwLineHeight, &ch, 1);
                    ReleaseDC(hwndMain, hdc);

                    // Store the character in the buffer.

                    pchInputBuf[cch++] = ch;

                    // Calculate the new horizontal position of the
                    // caret. If the new position exceeds the maximum,
                    // insert a carriage return and reposition the
                    // caret at the beginning of the next line.

                    nCaretPosX += nCharWidth;
                    if ((DWORD) nCaretPosX > dwMaxCharX)
                    {
                        nCaretPosX = 0;
                        pchInputBuf[cch++] = 0x0D;
                        ++nCurrentLine;
                    }
```

```
                ShowCaret(hwndMain);

                break;
        }
        SetCaretPos(nCaretPosX, nCurrentLine * dwLineHeight);
        nLastLine = max(nLastLine, nCurrentLine);
        break;

    // Process other messages.

    case WM_LBUTTONDOWN:

        // If a line of text is currently highlighted, redraw
        // the text to remove the highlighting.

        if (fTextSelected)
        {
            hdc = GetDC(hwndMain);
            SetTextColor(hdc, crPrevText);
            SetBkColor(hdc, crPrevBk);
            hResult = StringCchLength(szHilite, 128/sizeof(TCHAR), pcch);
            if (FAILED(hResult))
            {
            // TODO: write error handler
            }
            TextOut(hdc, 0, nCurrentLine * dwLineHeight,
                szHilite, *pcch);
            ReleaseDC(hwndMain, hdc);
            ShowCaret(hwndMain);
            fTextSelected = FALSE;
        }

        // Save the current mouse-cursor coordinates.

        ptsCursor = MAKEPOINTS(lParam);

        // Determine which line the cursor is on, and save
        // the line number. Do not allow line numbers greater
        // than the number of the last line of text. The
        // line number is later multiplied by the average height
        // of the current font. The result is used to set the
        // y-coordinate of the caret.

        nCurrentLine = min((int)(ptsCursor.y / dwLineHeight),
            nLastLine);

        // Parse the text input buffer to find the first
        // character in the selected line of text. Each
        // line ends with a carriage return, so it is possible
        // to count the carriage returns to find the selected
        // line.

        cCR = 0;
        nBegLine = 0;
        if (nCurrentLine != 0)
        {
            for (i = 0; (i < cch) &&
                    (cCR < nCurrentLine); i++)
            {
                if (pchInputBuf[i] == 0x0D)
                    ++cCR;
            }
            nBegLine = i;
        }

        // Starting at the beginning of the selected line,
        // measure  the width of each character, summing the
        // width with each character measured. Stop when the
        // sum is greater than the x-coordinate of the cursor.
```

```
            // sum is greater than the x-coordinate of the cursor.
            // The sum is used to set the x-coordinate of the caret.

            hdc = GetDC(hwndMain);
            nCaretPosX = 0;
            for (i = nBegLine;
                (pchInputBuf[i] != 0x0D) && (i < cch); i++)
            {
                ch = pchInputBuf[i];
                GetCharWidth32(hdc, (int) ch, (int) ch, &nCharWidth);
                if ((nCaretPosX + nCharWidth) > ptsCursor.x) break;
                else nCaretPosX += nCharWidth;
            }
            ReleaseDC(hwndMain, hdc);

            // Set the caret to the user-selected position.

            SetCaretPos(nCaretPosX, nCurrentLine * dwLineHeight);
            break;

        case WM_LBUTTONDBLCLK:

            // Copy the selected line of text to a buffer.

            for (i = nBegLine, j = 0; (pchInputBuf[i] != 0x0D) &&
                    (i < cch); i++)
            {
                szHilite[j++] = pchInputBuf[i];
            }
            szHilite[j] = '\0';

            // Hide the caret, invert the background and foreground
            // colors, and then redraw the selected line.

            HideCaret(hwndMain);
            hdc = GetDC(hwndMain);
            crPrevText = SetTextColor(hdc, RGB(255, 255, 255));
            crPrevBk = SetBkColor(hdc, RGB(0, 0, 0));
            hResult = StringCchLength(szHilite, 128/sizeof(TCHAR), pcch);
            if (FAILED(hResult))
            {
            // TODO: write error handler
            }
            TextOut(hdc, 0, nCurrentLine * dwLineHeight, szHilite, *pcch);
            SetTextColor(hdc, crPrevText);
            SetBkColor(hdc, crPrevBk);
            ReleaseDC(hwndMain, hdc);

            fTextSelected = TRUE;
            break;

            // Process other messages.

        default:
            return DefWindowProc(hwndMain, uMsg, wParam, lParam);
    }
    return NULL;
}
```

Usando uma roda do mouse em um documento com objetos inseridos

Este exemplo pressupõe um Microsoft Word com vários objetos inseridos:

- Uma planilha Microsoft Excel dados
- Um controle de caixa de listagem inserido que rola em resposta à roda

- Um controle de caixa de texto inserido que não responde à roda

A mensagem MSH _ MOUSEWHEEL sempre é enviada para a janela principal no Microsoft Word. Isso é verdadeiro mesmo se a planilha inserida estiver ativa. A tabela a seguir explica como a mensagem _ MSH MOUSEWHEEL é tratada de acordo com o foco.

O FOCO ESTÁ EM	A MANIPULAÇÃO É A SEGUINTE
Documento do Word	O Word rola a janela do documento.
Planilha Excel inserida	O Word posta a mensagem Excel. Você deve decidir se o aplicativo inserido deve responder à mensagem ou não.
Controle inserido	É responsabilidade do aplicativo enviar a mensagem para um controle inserido que tem o foco e verificar o código de retorno para ver se o controle o tratou. Se o controle não o tiver controlado, o aplicativo deverá rolar a janela do documento. Por exemplo, se o usuário clicasse em uma caixa de listagem e rolasse a roda, esse controle rolaria em resposta a uma rotação de roda. Se o usuário clicasse em uma caixa de texto e girasse a roda, o documento inteiro rolaria.

Este exemplo a seguir mostra como um aplicativo pode lidar com as mensagens de duas roda.

```
/************************************************
* this code deals with MSH_MOUSEWHEEL
************************************************/
#include "zmouse.h"

//
// Mouse Wheel rotation stuff, only define if we are
// on a version of the OS that does not support
// WM_MOUSEWHEEL messages.
//
#ifndef WM_MOUSEWHEEL
#define WM_MOUSEWHEEL WM_MOUSELAST+1
    // Message ID for IntelliMouse wheel
#endif

UINT uMSH_MOUSEWHEEL = 0;   // Value returned from
                            // RegisterWindowMessage()

/************************************************

INT WINAPI WinMain(
         HINSTANCE hInst,
         HINSTANCE hPrevInst,
         LPSTR lpCmdLine,
         INT nCmdShow)
{
    MSG msg;
    BOOL bRet;

    if (!InitInstance(hInst, nCmdShow))
        return FALSE;

    //
    // The new IntelliMouse uses a Registered message to transmit
    // wheel rotation info. So register for it!

    uMSH_MOUSEWHEEL =
     RegisterWindowMessage(MSH_MOUSEWHEEL);
    if ( !uMSH_MOUSEWHEEL )
```

```
    if ( !uMSH_MOUSEWHEEL )
    {
        MessageBox(NULL,"
                   RegisterWindowMessag Failed!",
                   "Error",MB_OK);
        return msg.wParam;
    }

    while (( bRet = GetMessage(&msg, NULL, 0, 0)) != 0)
    {
        if (bRet == -1)
        {
            // handle the error and possibly exit
        }
        else
        {
            if (!TranslateAccelerator(ghwndApp,
                                      ghaccelTable,
                                      &msg))
            {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
            }
        }
    }

    return msg.wParam;
}

/*********************************************
* this code deals with WM_MOUSEWHEEL
*********************************************/
LONG APIENTRY MainWndProc(
    HWND hwnd,
    UINT msg,
    WPARAM wParam,
    LPARAM lParam)
{
    static int nZoom = 0;


    switch (msg)
    {
        //
        // Handle Mouse Wheel messages generated
        // by the operating systems that have built-in
        // support for the WM_MOUSEWHEEL message.
        //

        case WM_MOUSEWHEEL:
            ((short) HIWORD(wParam)< 0) ? nZoom-- : nZoom++;

            //
            // Do other wheel stuff...
            //

            break;

        default:
            //
            // uMSH_MOUSEWHEEL is a message registered by
            // the mswheel dll on versions of Windows that
            // do not support the new message in the OS.

            if( msg == uMSH_MOUSEWHEEL )
            {
               ((int)wParam < 0) ? nZoom-- : nZoom++;

                //
```

```
                // Do other wheel stuff...
                //
                break;
            }

            return DefWindowProc(hwnd,
                                 msg,
                                 wParam,
                                 lParam);
    }

    return 0L;
}
```

Recuperando o número de linhas de rolagem da roda do mouse

O código a seguir permite que um aplicativo recupere o número de linhas de rolagem usando a **função SystemParametersInfo.**

```
#ifndef SPI_GETWHEELSCROLLLINES
#define SPI_GETWHEELSCROLLLINES   104
#endif

#include "zmouse.h"

/*********************************************************
* FUNCTION: GetNumScrollLines
* Purpose : An OS independent method to retrieve the
*           number of wheel scroll lines
* Params  : none
* Returns : UINT:  Number of scroll lines where WHEEL_PAGESCROLL
*           indicates to scroll a page at a time.
*********************************************************/
UINT GetNumScrollLines(void)
{
   HWND hdlMsWheel;
   UINT ucNumLines=3;  // 3 is the default
   OSVERSIONINFO osversion;
   UINT uiMsh_MsgScrollLines;


   memset(&osversion, 0, sizeof(osversion));
   osversion.dwOSVersionInfoSize =sizeof(osversion);
   GetVersionEx(&osversion);

   if ((osversion.dwPlatformId == VER_PLATFORM_WIN32_WINDOWS) ||
       ( (osversion.dwPlatformId == VER_PLATFORM_WIN32_NT) &&
         (osversion.dwMajorVersion < 4) )   )
   {
        hdlMsWheel = FindWindow(MSH_WHEELMODULE_CLASS,
                                MSH_WHEELMODULE_TITLE);
        if (hdlMsWheel)
        {
           uiMsh_MsgScrollLines = RegisterWindowMessage
                                    (MSH_SCROLL_LINES);
           if (uiMsh_MsgScrollLines)
                ucNumLines = (int)SendMessage(hdlMsWheel,
                                      uiMsh_MsgScrollLines,
                                                      0,
                                                      0);
        }
   }
   else if ( (osversion.dwPlatformId ==
                         VER_PLATFORM_WIN32_NT) &&
             (osversion.dwMajorVersion >= 4) )
   {
      SystemParametersInfo(SPI_GETWHEELSCROLLLINES,
                                         0,
                                      &ucNumLines, 0);
   }
   return(ucNumLines);
}
```

Referência de entrada do mouse

15/04/2022 • 2 minutes to read

Os tópicos contidos nesta seção fornecem as especificações de referência para entrada do mouse.

Nesta seção

TÓPICO	DESCRIÇÃO
Funções de entrada do mouse	
Macros de entrada do mouse	
Notificações de entrada do mouse	
Estruturas de entrada do mouse	

Funções de entrada do mouse

15/04/2022 • 2 minutes to read

Nesta seção

TÓPICO	DESCRIÇÃO
_TrackMouseEvent	Poste mensagens quando o ponteiro do mouse sair de uma janela ou passar o mouse sobre uma janela por um período especificado. Essa função chamará TrackMouseEvent se ele existir, caso contrário, ele o emula.
DragDetect	Captura o mouse e acompanha seu movimento até que o usuário libere o botão esquerdo, pressione a tecla ESC ou mova o mouse para fora do retângulo de arrastar ao redor do ponto especificado. A largura e a altura do retângulo de arrastar são especificadas pelos valores **SM_CXDRAG** e SM_CYDRAG retornados pela função GetSystemMetrics.
GetCapture	Recupera um alça para a janela (se há) que capturou o mouse. Somente uma janela por vez pode capturar o mouse; essa janela recebe a entrada do mouse se o cursor está ou não dentro de suas bordas.
GetDoubleClickTime	Recupera a hora atual de clique duplo para o mouse. Um clique duplo é uma série de dois cliques do botão do mouse, o segundo ocorrendo dentro de um horário especificado após o primeiro. O tempo de clique duplo é o número máximo de milissegundos que podem ocorrer entre o primeiro e o segundo clique de um clique duplo. O tempo máximo de clique duplo é de 5.000 milissegundos.
GetMouseMovePointsEx	Recupera um histórico de até 64 coordenadas anteriores do mouse ou caneta.
mouse_event	A mouse_event sintetiza o movimento do mouse e os cliques de botão. [!Note] Essa função foi superada. Em vez disso, use SendInput.
Releasecapture	Libera a captura do mouse de uma janela no thread atual e restaura o processamento normal de entrada do mouse. Uma janela que capturou o mouse recebe toda a entrada do mouse, independentemente da posição do cursor, exceto quando um botão do mouse é clicado enquanto o cursor está na janela de outro thread.
Setcapture	Define a captura do mouse para a janela especificada que pertence ao thread atual.

TÓPICO	DESCRIÇÃO
SetDoubleClickTime	Define a hora de clique duplo para o mouse. Um clique duplo é uma série de dois cliques de um botão do mouse, o segundo ocorrendo dentro de um horário especificado após o primeiro. O tempo de clique duplo é o número máximo de milissegundos que podem ocorrer entre o primeiro e o segundo cliques de um clique duplo.
SwapMouseButton	Inverte ou restaura o significado dos botões esquerdo e direito do mouse.
Trackmouseevent	Poste mensagens quando o ponteiro do mouse sair de uma janela ou passar o mouse sobre uma janela por um período especificado. [!Note] A _TrackMouseEvent chamada TrackMouseEvent se ela existir, caso contrário, _TrackMouseEvent **emula TrackMouseEvent.**

Macros de entrada do mouse

15/04/2022 • 2 minutes to read

Nesta seção

- OBTER _ _ lParam APPCOMMAND
- OBTER _ lParam do dispositivo _
- OBTER os _ sinalizadores _ lParam
- OBTER _ _ lParam KeyState
- OBTER _ _ wParam KeyState
- OBTER _ _ lParam MOUSEORKEY
- OBTER _ _ wParam NCHITTEST
- OBTER _ _ wParam Delta do volante _
- OBTER _ _ wParam XBUTTON

Notificações de entrada do mouse

15/04/2022 • 2 minutes to read

Nesta seção

- WM _ capturachanged
- LBUTTONDBLCLK do WM _
- LBUTTONDOWN do WM _
- LBUTTONUP do WM _
- MBUTTONDBLCLK do WM _
- MBUTTONDOWN do WM _
- MBUTTONUP do WM _
- MOUSEACTIVATE do WM _
- MOUSEHOVER do WM _
- MOUSEHWHEEL do WM _
- MOUSELEAVE do WM _
- admousemove do WM _
- MOUSEWHEEL do WM _
- NCHITTEST do WM _
- NCLBUTTONDBLCLK do WM _
- NCLBUTTONDOWN do WM _
- NCLBUTTONUP do WM _
- NCMBUTTONDBLCLK do WM _
- NCMBUTTONDOWN do WM _
- NCMBUTTONUP do WM _
- NCMOUSEHOVER do WM _
- NCMOUSELEAVE do WM _
- NCMOUSEMOVE do WM _
- NCRBUTTONDBLCLK do WM _
- NCRBUTTONDOWN do WM _
- NCRBUTTONUP do WM _
- NCXBUTTONDBLCLK do WM _
- NCXBUTTONDOWN do WM _
- NCXBUTTONUP do WM _
- RBUTTONDBLCLK do WM _
- RBUTTONDOWN do WM _
- RBUTTONUP do WM _
- XBUTTONDBLCLK do WM _
- XBUTTONDOWN do WM _
- XBUTTONUP do WM _

Mensagem do WM _ capturachanged

15/04/2022 • 2 minutes to read

Enviado para a janela que está perdendo a captura do mouse.

Uma janela recebe essa mensagem por meio de sua função **WindowProc** .

```
#define WM_CAPTURECHANGED             0x0215
```

Parâmetros

wParam

Este parâmetro não é usado.

lParam

Um identificador para a janela que ganha a captura do mouse.

Retornar valor

Um aplicativo deve retornar zero se ele processar essa mensagem.

Comentários

Uma janela recebe essa mensagem, mesmo que chame o próprio **ReleaseCapture** . Um aplicativo não deve tentar definir a captura do mouse em resposta a essa mensagem.

Quando ele recebe essa mensagem, uma janela deve ser redesenhada, se necessário, para refletir o novo estado de captura do mouse.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser. h (incluir Windows. h)

Confira também

Referência

ReleaseCapture

SetCapture

Conceitual

Entrada do mouse

Mensagem WM _ LBUTTONDBLCLK

15/04/2022 • 2 minutes to read

Postado quando o usuário clica duas vezes no botão esquerdo do mouse enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.

Uma janela recebe essa mensagem por meio de **sua função WindowProc.**

```
#define WM_LBUTTONDBLCLK                0x0203
```

Parâmetros

wParam

Indica se várias chaves virtuais estão inoadas. Esse parâmetro pode ser um ou mais dos valores a seguir.

VALOR	SIGNIFICADO
MK _ CONTROLE 0x0008	A tecla CTRL está inocizada.
MK _ LBUTTON 0x0001	O botão esquerdo do mouse está ino mouse.
MK _ MBUTTON 0x0010	O botão do meio do mouse está ino mouse.
MK _ RBUTTON 0x0002	O botão direito do mouse está ino mouse.
MK _ SHIFT 0x0004	A tecla SHIFT está inobada.
MK _ XBUTTON1 0x0020	O primeiro botão X está ino mouse.
MK _ XBUTTON2 0x0040	O segundo botão X está ino mouse.

lParam

A palavra de ordem baixa especifica a coordenada X do cursor. A coordenada é relativa ao canto superior

esquerdo da área do cliente.

A palavra de ordem alta especifica a coordenada y do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

Retornar valor

Se um aplicativo processa essa mensagem, ele deve retornar zero.

Comentários

Use o código a seguir para obter a posição horizontal e vertical:

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

Conforme mencionado acima, a coordenada x está na ordem baixa **a menos** que o valor de retorno; a coordenada y está em ordem alta curta **(ambos** representam valores assinados porque podem receber valores negativos em sistemas com vários monitores). Se o valor de retorno for atribuído a uma variável, você poderá usar a macro MAKEPOINTS para obter uma estrutura POINTS do valor de retorno. Você também pode usar a macro GET _ X _ LPARAM ou GET Y _ _ LPARAM para extrair a coordenada x ou y.

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor porque essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas, e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades sem sinal.

Somente as janelas que têm o estilo _ **DBLCLKS** do CS podem receber mensagens **WM _ LBUTTONDBLCLK,** que o sistema gera sempre que o usuário pressiona, libera e pressiona novamente o botão esquerdo do mouse dentro do limite de tempo de clique duplo do sistema. Clicar duas vezes no botão esquerdo do mouse gera uma sequência de quatro mensagens: WM _ LBUTTONDOWN, WM _ LBUTTONUP, **WM _ LBUTTONDBLCLK** e **WM _ LBUTTONUP.**

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (inclua Windowsx.h)

Confira também

Referência

GET _ X _ LPARAM

GET _ Y _ LPARAM

GetCapture

GetDoubleClickTime

Setcapture

SetDoubleClickTime

WM _ LBUTTONDOWN

WM _ LBUTTONUP

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

PONTOS

Mensagem do WM _ LBUTTONDOWN

15/04/2022 • 2 minutes to read

Postado quando o usuário pressiona o botão esquerdo do mouse enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.

Uma janela recebe essa mensagem por meio de sua função **WindowProc** .

```
#define WM_LBUTTONDOWN                  0x0201
```

Parâmetros

wParam

Indica se várias chaves virtuais estão inativas. Esse parâmetro pode ser um ou mais dos valores a seguir.

VALOR	SIGNIFICADO
MK _ 0X0008 de controle	A tecla CTRL está inoperante.
MK _ LBUTTON 0x0001	O botão esquerdo do mouse está inativo.
MK _ MBUTTON 0x0010	O botão do meio do mouse está inativo.
MK _ RBUTTON 0x0002	O botão direito do mouse está inativo.
MK _ SHIFT 0x0004	A tecla SHIFT está inoperante.
MK _ XBUTTON1 0x0020	O primeiro botão X está inoperante.
MK _ XBUTTON2 0x0040	O segundo botão X está inoperante.

lParam

A palavra de ordem inferior Especifica a coordenada x do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

A palavra de ordem superior especifica a coordenada y do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

Retornar valor

Se um aplicativo processar essa mensagem, ele deverá retornar zero.

Exemplo

```
LRESULT CALLBACK WndProc(_In_ HWND hWnd, _In_ UINT msg, _In_ WPARAM wParam, _In_ LPARAM lParam)
{
    POINT pt;

    switch (msg)
    {

    case WM_LBUTTONDOWN:
            {
                pt.x = GET_X_LPARAM(lParam);
                pt.y = GET_Y_LPARAM(lParam);
            }
        }
        break;

    default:
        return DefWindowProc(hWnd, msg, wParam, lParam);
    }
    return 0;
}
```

para obter mais exemplos, consulte Windows exemplos clássicos em GitHub.

Comentários

Conforme observado acima, a coordenada x está **na ordem inferior** do valor de retorno; a coordenada y está no **intervalo** de ordem superior (ambos representam valores *assinados* porque podem receber valores negativos em sistemas com vários monitores). Se o valor de retorno for atribuído a uma variável, você poderá usar a macro **MAKEPOINTS** para obter uma estrutura de **pontos** do valor de retorno. Você também pode usar a macro **Get _ X _ lParam** ou **Get _ y _ lParam** para extrair a coordenada x ou Y.

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor, pois essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades não assinadas.

Para detectar que a tecla ALT foi pressionada, verifique se **GetKeyState** com o **_ menu VK** < 0. Observe que isso não deve ser **GetAsyncKeyState**.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	WinUser. h (incluir Windowsx. h)

Confira também

Referência

OBTER _ X _ lParam

OBTER _ _ lParam Y

GetCapture

GetKeyState

SetCapture

LBUTTONDBLCLK do WM _

LBUTTONUP do WM _

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

FAIXAS

Mensagem do WM _ LBUTTONUP

15/04/2022 • 2 minutes to read

Postado quando o usuário libera o botão esquerdo do mouse enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.

Uma janela recebe essa mensagem por meio de sua função **WindowProc** .

```
#define WM_LBUTTONUP                    0x0202
```

Parâmetros

wParam

Indica se várias chaves virtuais estão inativas. Esse parâmetro pode ser um ou mais dos valores a seguir.

VALOR	SIGNIFICADO
MK _ 0X0008 de controle	A tecla CTRL está inoperante.
MK _ MBUTTON 0x0010	O botão do meio do mouse está inativo.
MK _ RBUTTON 0x0002	O botão direito do mouse está inativo.
MK _ SHIFT 0x0004	A tecla SHIFT está inoperante.
MK _ XBUTTON1 0x0020	O primeiro botão X está inoperante.
MK _ XBUTTON2 0x0040	O segundo botão X está inoperante.

lParam

A palavra de ordem inferior Especifica a coordenada x do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

A palavra de ordem superior especifica a coordenada y do cursor. A coordenada é relativa ao canto superior

esquerdo da área do cliente.

Retornar valor

Se um aplicativo processar essa mensagem, ele deverá retornar zero.

Comentários

Use o código a seguir para obter a posição horizontal e vertical:

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

Conforme observado acima, a coordenada x está no **intervalo** de ordem inferior do valor de lParam; a coordenada y está no **intervalo** de ordem superior (ambos representam valores *assinados* porque podem receber valores negativos em sistemas com vários monitores). Se o valor de retorno for atribuído a uma variável, você poderá usar a macro **MAKEPOINTS** para obter uma estrutura de **pontos** do valor de retorno. Você também pode usar a macro **Get _ X _ lParam** ou **Get _ y _ lParam** para extrair a coordenada x ou Y.

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor, pois essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades não assinadas.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	WinUser. h (incluir Windowsx. h)

Confira também

Referência

OBTER _ X _ lParam

OBTER _ _ lParam Y

GetCapture

SetCapture

LBUTTONDBLCLK do WM _

LBUTTONDOWN do WM _

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

FAIXAS

Mensagem do WM _ MBUTTONDBLCLK

15/04/2022 • 2 minutes to read

Postado quando o usuário clica duas vezes no botão do meio do mouse enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.

Uma janela recebe essa mensagem por meio de sua função **WindowProc** .

```
#define WM_MBUTTONDBLCLK                0x0209
```

Parâmetros

wParam

Indica se várias chaves virtuais estão inativas. Esse parâmetro pode ser um ou mais dos valores a seguir.

VALOR	SIGNIFICADO
MK _ 0X0008 de controle	A tecla CTRL está inoperante.
MK _ LBUTTON 0x0001	O botão esquerdo do mouse está inativo.
MK _ MBUTTON 0x0010	O botão do meio do mouse está inativo.
MK _ RBUTTON 0x0002	O botão direito do mouse está inativo.
MK _ SHIFT 0x0004	A tecla SHIFT está inoperante.
MK _ XBUTTON1 0x0020	O primeiro botão X está inoperante.
MK _ XBUTTON2 0x0040	O segundo botão X está inoperante.

lParam

A palavra de ordem inferior Especifica a coordenada x do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

A palavra de ordem superior especifica a coordenada y do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

Retornar valor

Se um aplicativo processar essa mensagem, ele deverá retornar zero.

Comentários

Use o código a seguir para obter a posição horizontal e vertical:

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

Conforme observado acima, a coordenada x está **na ordem inferior** do valor de retorno; a coordenada y está no **intervalo** de ordem superior (ambos representam valores *assinados* porque podem receber valores negativos em sistemas com vários monitores). Se o valor de retorno for atribuído a uma variável, você poderá usar a macro MAKEPOINTS para obter uma estrutura de **pontos** do valor de retorno. Você também pode usar a macro Get _ X _ lParam ou Get _ y _ lParam para extrair a coordenada x ou Y.

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor, pois essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades não assinadas.

Somente as janelas que têm o estilo **cs _ DBLCLKS** podem receber mensagens do **WM _ MBUTTONDBLCLK** , que o sistema gera quando o usuário pressiona, libera e novamente pressiona o botão do meio do mouse dentro do limite de tempo de clique duplo do sistema. Clicar duas vezes no botão do meio do mouse gera quatro mensagens : _ WM MBUTTONDOWN, WM _ MBUTTONUP, **WM _ MBUTTONDBLCLK** e **WM _ MBUTTONUP** novamente.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	WinUser. h (incluir Windowsx. h)

Confira também

Referência

OBTER _ X _ lParam

OBTER _ _ lParam Y

GetCapture

GetDoubleClickTime

SetCapture

Setdoubleclicktime

MBUTTONDOWN do WM _

MBUTTONUP do WM _

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

FAIXAS

Mensagem do WM _ MBUTTONDOWN

15/04/2022 • 2 minutes to read

Postado quando o usuário pressiona o botão do meio do mouse enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.

Uma janela recebe essa mensagem por meio de sua função **WindowProc** .

```
#define WM_MBUTTONDOWN                  0x0207
```

Parâmetros

wParam

Indica se várias chaves virtuais estão inativas. Esse parâmetro pode ser um ou mais dos valores a seguir.

VALOR	SIGNIFICADO
MK _ 0X0008 de controle	A tecla CTRL está inoperante.
MK _ LBUTTON 0x0001	O botão esquerdo do mouse está inativo.
MK _ MBUTTON 0x0010	O botão do meio do mouse está inativo.
MK _ RBUTTON 0x0002	O botão direito do mouse está inativo.
MK _ SHIFT 0x0004	A tecla SHIFT está inoperante.
MK _ XBUTTON1 0x0020	O primeiro botão X está inoperante.
MK _ XBUTTON2 0x0040	O segundo botão X está inoperante.

lParam

A palavra de ordem inferior Especifica a coordenada x do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

A palavra de ordem superior especifica a coordenada y do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

Retornar valor

Se um aplicativo processar essa mensagem, ele deverá retornar zero.

Comentários

Use o código a seguir para obter a posição horizontal e vertical:

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

Conforme observado acima, a coordenada x está **na ordem inferior** do valor de retorno; a coordenada y está no **intervalo** de ordem superior (ambos representam valores *assinados* porque podem receber valores negativos em sistemas com vários monitores). Se o valor de retorno for atribuído a uma variável, você poderá usar a macro MAKEPOINTS para obter uma estrutura de **pontos** do valor de retorno. Você também pode usar a macro Get _ X _ lParam ou Get _ y _ lParam para extrair a coordenada x ou Y.

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor, pois essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades não assinadas.

Para detectar que a tecla ALT foi pressionada, verifique se GetKeyState com o **_ menu VK** < 0. Observe que isso não deve ser GetAsyncKeyState.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	WinUser. h (incluir Windowsx. h)

Confira também

Referência

OBTER _ X _ lParam

OBTER _ _ lParam Y

GetCapture

GetKeyState

SetCapture

MBUTTONDBLCLK do WM _

MBUTTONUP do WM _

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

FAIXAS

Mensagem _ WM MBUTTONUP

15/04/2022 • 2 minutes to read

Postado quando o usuário libera o botão do meio do mouse enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.

Uma janela recebe essa mensagem por meio de **sua função WindowProc.**

```
#define WM_MBUTTONUP                    0x0208
```

Parâmetros

wParam

Indica se várias chaves virtuais estão inoadas. Esse parâmetro pode ser um ou mais dos valores a seguir.

VALOR	SIGNIFICADO
MK _ CONTROLE 0x0008	A tecla CTRL está inocizada.
MK _ LBUTTON 0x0001	O botão esquerdo do mouse está ino mouse.
MK _ MBUTTON 0x0010	O botão do meio do mouse está ino mouse.
MK _ RBUTTON 0x0002	O botão direito do mouse está ino mouse.
MK _ SHIFT 0x0004	A tecla SHIFT está inobada.
MK _ XBUTTON1 0x0020	O primeiro botão X está ino mouse.
MK _ XBUTTON2 0x0040	O segundo botão X está ino mouse.

lParam

A palavra de ordem baixa especifica a coordenada X do cursor. A coordenada é relativa ao canto superior

esquerdo da área do cliente.

A palavra de ordem alta especifica a coordenada y do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

Observe que quando um menu de atalho está presente (exibido), as coordenadas são relativas à tela, não à área do cliente. Como TrackPopupMenu é uma chamada assíncrona e a notificação **wm _ MBUTTONUP** não tem um sinalizador especial que indica derivação de coordenadas, um aplicativo não pode dizer se as coordenadas x,y contidas em *lParam* são relativas à tela ou à área do cliente.

Retornar valor

Se um aplicativo processa essa mensagem, ele deve retornar zero.

Comentários

Use o código a seguir para obter a posição horizontal e vertical:

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

Conforme mencionado acima, a coordenada x está na ordem baixa **a menos** que o valor de retorno; a coordenada y está em ordem alta curta **(ambos** representam valores assinados porque podem receber valores negativos em sistemas com vários monitores). Se o valor de retorno for atribuído a uma variável, você poderá usar a macro MAKEPOINTS para obter uma estrutura POINTS do valor de retorno. Você também pode usar a macro GET _ X _ LPARAM ou GET Y _ _ LPARAM para extrair a coordenada x ou y.

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor porque essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas, e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades sem sinal.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (inclua Windowsx.h)

Confira também

Referência

GET _ X _ LPARAM

GET _ Y _ LPARAM

GetCapture

Setcapture

WM _ MBUTTONDBLCLK

WM _ MBUTTONDOWN

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

PONTOS

Mensagem WM _ MOUSEACTIVATE

15/04/2022 • 2 minutes to read

Enviado quando o cursor está em uma janela inativa e o usuário pressiona um botão do mouse. A janela pai receberá essa mensagem somente se a janela filho passá-la para a **função DefWindowProc.**

Uma janela recebe essa mensagem por meio de **sua função WindowProc.**

```
#define WM_MOUSEACTIVATE                0x0021
```

Parâmetros

wParam

Um alça para a janela pai de nível superior da janela que está sendo ativada.

lParam

A palavra de ordem baixa especifica o valor de teste de acerto retornado pela função **DefWindowProc** como resultado do processamento da mensagem **WM _ NCHITTEST.** Para ver uma lista de valores de teste de acerto, consulte **WM _ NCHITTEST**.

A palavra de ordem alta especifica o identificador da mensagem do mouse gerada quando o usuário pressionou um botão do mouse. A mensagem do mouse é descartada ou postada na janela, dependendo do valor de retorno.

Retornar valor

O valor de retorno especifica se a janela deve ser ativada e se o identificador da mensagem do mouse deve ser descartado. Ele deve ser um dos valores a seguir.

VALOR/CÓDIGO DE RETORNO	DESCRIÇÃO
MA _ ATIVAR 1	Ativa a janela e não descarta a mensagem do mouse.
MA _ ACTIVATEANDEAT 2	Ativa a janela e descarta a mensagem do mouse.
MA _ NOACTIVATE 3	Não ativa a janela e não descarta a mensagem do mouse.
MA _ NOACTIVATEANDEAT 4	Não ativa a janela, mas descarta a mensagem do mouse.

Comentários

A **função DefWindowProc** passa a mensagem para a janela pai de uma janela filho antes que qualquer processamento ocorra. A janela pai determina se a janela filho deve ser ativada. Se ela ativar a janela filho, a janela pai deverá retornar **MA _ NOACTIVATE** ou **MA _ NOACTIVATEANDEAT** para impedir que o sistema processe ainda mais a mensagem.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (incluir Windows.h)

Confira também

Referência

Defwindowproc

HIWORD

LOWORD

WM _ NCHITTEST

Conceitual

Entrada do mouse

Mensagem _ WM MOUSEHOVER

15/04/2022 • 2 minutes to read

Postado em uma janela quando o cursor passar o mouse sobre a área do cliente da janela pelo período de tempo especificado em uma chamada anterior para **TrackMouseEvent**.

Uma janela recebe essa mensagem por meio de **sua função WindowProc.**

```
#define WM_MOUSEHOVER                   0x02A1
```

Parâmetros

wParam

Indica se várias chaves virtuais estão inoadas. Esse parâmetro pode ser um ou mais dos valores a seguir.

VALOR	SIGNIFICADO
MK _ CONTROLE 0x0008	A tecla CTRL está desaixado.
MK _ LBUTTON 0x0001	O botão esquerdo do mouse está pressionado.
MK _ MBUTTON 0x0010	O botão do meio do mouse é pressionado.
MK _ RBUTTON 0x0002	O botão direito do mouse está pressionado.
MK _ SHIFT 0x0004	A tecla SHIFT é desaixado.
MK _ XBUTTON1 0x0020	O primeiro botão X está ino mouse.
MK _ XBUTTON2 0x0040	O segundo botão X está ino mouse.

lParam

A palavra de ordem baixa especifica a coordenada X do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

A palavra de ordem alta especifica a coordenada y do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

Retornar valor

Se um aplicativo processa essa mensagem, ele deve retornar zero.

Comentários

O acompanhamento de foco para **quando WM _ MOUSEHOVER** é gerado. O aplicativo deverá chamar TrackMouseEvent novamente se exigir um acompanhamento posterior do comportamento de foco do mouse.

Use o código a seguir para obter a posição horizontal e vertical:

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

Conforme mencionado acima, a coordenada x está na ordem baixa **a menos** que o valor de retorno; a coordenada y está em ordem alta curta **(ambos** representam valores assinados porque podem receber valores negativos em sistemas com vários monitores). Se o valor de retorno for atribuído a uma variável, você poderá usar a macro MAKEPOINTS para obter uma estrutura POINTS do valor de retorno. Você também pode usar a macro GET _ X _ LPARAM ou GET Y _ _ LPARAM para extrair a coordenada x ou y.

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor porque essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas, e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades sem sinal.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (inclua Windowsx.h)

Confira também

Referência

GET _ X _ LPARAM

GET _ Y _ LPARAM

GetCapture

Setcapture

Trackmouseevent

TRACKMOUSEEVENT

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

PONTOS

Mensagem _ WM MOUSEHWHEEL

15/04/2022 • 3 minutes to read

Enviado para a janela ativa quando a roda de rolagem horizontal do mouse é inclinada ou girada. A **função DefWindowProc** propaga a mensagem para o pai da janela. Não deve haver encaminhamento interno da mensagem, pois **DefWindowProc** a propaga para cima na cadeia pai até encontrar uma janela que a processe.

Uma janela recebe essa mensagem por meio de **sua função WindowProc.**

```
#define WM_MOUSEHWHEEL                  0x020E
```

Parâmetros

wParam

A palavra de ordem alta indica a distância em que a roda é girada, expressa em múltiplos ou fatores de **WHEEL _ DELTA,** que é definido como 120. Um valor positivo indica que a roda foi girada para a direita; um valor negativo indica que a roda foi girada para a esquerda.

A palavra de ordem baixa indica se várias chaves virtuais estão inotivas. Esse parâmetro pode ser um ou mais dos valores a seguir.

VALOR	SIGNIFICADO
MK _ CONTROLE 0x0008	A tecla CTRL está inocizada.
MK _ LBUTTON 0x0001	O botão esquerdo do mouse está ino mouse.
MK _ MBUTTON 0x0010	O botão do meio do mouse está ino mouse.
MK _ RBUTTON 0x0002	O botão direito do mouse está ino mouse.
MK _ SHIFT 0x0004	A tecla SHIFT está inobada.
MK _ XBUTTON1 0x0020	O primeiro botão X está ino mouse.

VALOR	SIGNIFICADO
MK _ XBUTTON2 0x0040	O segundo botão X está ino mouse.

lParam

A palavra de ordem baixa especifica a coordenada X do ponteiro em relação ao canto superior esquerdo da tela.

A palavra de ordem alta especifica a coordenada y do ponteiro em relação ao canto superior esquerdo da tela.

Retornar valor

Se um aplicativo processa essa mensagem, ele deve retornar zero.

Comentários

Use o código a seguir para obter as informações no *parâmetro wParam.*

```
fwKeys = GET_KEYSTATE_WPARAM(wParam);
zDelta = GET_WHEEL_DELTA_WPARAM(wParam);
```

Use o código a seguir para obter a posição horizontal e vertical.

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

Conforme mencionado acima, a coordenada x está na ordem baixa **a menos** que o valor de retorno; a coordenada y está em ordem alta curta **(ambos** representam valores assinados porque podem receber valores negativos em sistemas com vários monitores). Se o valor de retorno for atribuído a uma variável, você poderá usar a macro MAKEPOINTS para obter uma estrutura POINTS do valor de retorno. Você também pode usar a macro GET _ X _ LPARAM ou GET Y _ _ LPARAM para extrair a coordenada x ou y.

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor porque essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas, e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades sem sinal.

A rotação de roda é um múltiplo **de WHEEL _ DELTA,** que é definido como 120. Esse é o limite para a ação a ser tomada e uma dessas ações (por exemplo, rolar um incremento) deve ocorrer para cada delta.

O delta foi definido como 120 para permitir que a Microsoft ou outros fornecedores criem roda de resolução mais fina (por exemplo, uma roda giratória livremente sem entalhes) para enviar mais mensagens por rotação, mas com um valor menor em cada mensagem. Para usar esse recurso, você pode adicionar os valores delta de entrada até que **WHEEL _ DELTA** seja atingido (portanto, para uma rotação delta você obter a mesma resposta) ou rolar linhas parciais em resposta a mensagens mais frequentes. Você também pode escolher a granularidade de rolagem e acumular deltas até que ela seja atingida.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows Somente [aplicativos da área de trabalho do Vista]
Servidor mínimo com suporte	Windows Somente aplicativos da área de trabalho server 2008 []
Cabeçalho	Winuser.h (inclua Windowsx.h)

Confira também

Referência

GET _ KEYSTATE _ WPARAM

GET _ X _ LPARAM

GET _ Y _ LPARAM

GET _ WHEEL _ DELTA _ WPARAM

HIWORD

LOWORD

evento _ mouse

Conceitual

Entrada do mouse

Outros recursos

Getsystemmetrics

MAKEPOINTS

PONTOS

Systemparametersinfo

Mensagem de MOUSELEAVE do WM _

15/04/2022 • 2 minutes to read

Postado em uma janela quando o cursor sai da área do cliente da janela especificada em uma chamada anterior para **TrackMouseEvent**.

Uma janela recebe essa mensagem por meio de sua função **WindowProc** .

```
#define WM_MOUSELEAVE                   0x02A3
```

Parâmetros

wParam

Esse parâmetro não é usado e deve ser zero.

lParam

Esse parâmetro não é usado e deve ser zero.

Retornar valor

Se um aplicativo processar essa mensagem, ele deverá retornar zero.

Comentários

Todo o controle solicitado por **TrackMouseEvent** é cancelado quando esta mensagem é gerada. O aplicativo deve chamar **TrackMouseEvent** quando o mouse reinserir sua janela se precisar de mais controle sobre o comportamento de focalização do mouse.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser. h (incluir Windows. h)

Confira também

Referência

GetCapture

SetCapture

TrackMouseEvent

TRACKMOUSEEVENT

NCMOUSELEAVE do WM _

Conceitual

Entrada do mouse

_Mensagem MOUSEMOVE do WM

15/04/2022 • 2 minutes to read

Postado em uma janela quando o cursor se move. Se o mouse não for capturado, a mensagem será postada na janela que contém o cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.

Uma janela recebe essa mensagem por meio de sua função **WindowProc** .

```
#define WM_MOUSEMOVE                    0x0200
```

Parâmetros

wParam

Indica se várias chaves virtuais estão inativas. Esse parâmetro pode ser um ou mais dos valores a seguir.

VALOR	SIGNIFICADO
MK _ 0X0008 de controle	A tecla CTRL está inoperante.
MK _ LBUTTON 0x0001	O botão esquerdo do mouse está inativo.
MK _ MBUTTON 0x0010	O botão do meio do mouse está inativo.
MK _ RBUTTON 0x0002	O botão direito do mouse está inativo.
MK _ SHIFT 0x0004	A tecla SHIFT está inoperante.
MK _ XBUTTON1 0x0020	O primeiro botão X está inoperante.
MK _ XBUTTON2 0x0040	O segundo botão X está inoperante.

lParam

A palavra de ordem inferior Especifica a coordenada x do cursor. A coordenada é relativa ao canto superior

esquerdo da área do cliente.

A palavra de ordem superior especifica a coordenada y do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

Retornar valor

Se um aplicativo processar essa mensagem, ele deverá retornar zero.

Comentários

Use o código a seguir para obter a posição horizontal e vertical:

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

Conforme observado acima, a coordenada x está **na ordem inferior** do valor de retorno; a coordenada y está no **intervalo** de ordem superior (ambos representam valores *assinados* porque podem receber valores negativos em sistemas com vários monitores). Se o valor de retorno for atribuído a uma variável, você poderá usar a macro MAKEPOINTS para obter uma estrutura de **pontos** do valor de retorno. Você também pode usar a macro Get _ X _ lParam ou Get _ y _ lParam para extrair a coordenada x ou Y.

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor, pois essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades não assinadas.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	WinUser. h (incluir Windowsx. h)

Confira também

Referência

OBTER _ X _ lParam

OBTER _ _ lParam Y

GetCapture

SetCapture

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

FAIXAS

Mensagem do WM _ MOUSEWHEEL

15/04/2022 • 3 minutes to read

Enviado para a janela de foco quando a roda do mouse é girada. A função **DefWindowProc** propaga a mensagem para o pai da janela. Não deve haver nenhum encaminhamento interno da mensagem, pois **DefWindowProc** propaga a cadeia pai até encontrar uma janela que a processa.

Uma janela recebe essa mensagem por meio de sua função **WindowProc** .

```
#define WM_MOUSEWHEEL                   0x020A
```

Parâmetros

wParam

A palavra de ordem superior indica a distância em que a roda é girada, expressa em múltiplos ou divisões do **Wheel _ Delta**, que é 120. Um valor positivo indica que a roda foi girada para frente, afastando-a do usuário; um valor negativo indica que a roda foi girada para trás, em direção ao usuário.

A palavra de ordem inferior indica se várias chaves virtuais estão inativas. Esse parâmetro pode ser um ou mais dos valores a seguir.

VALOR	SIGNIFICADO
MK _ 0X0008 de controle	A tecla CTRL está inoperante.
MK _ LBUTTON 0x0001	O botão esquerdo do mouse está inativo.
MK _ MBUTTON 0x0010	O botão do meio do mouse está inativo.
MK _ RBUTTON 0x0002	O botão direito do mouse está inativo.
MK _ SHIFT 0x0004	A tecla SHIFT está inoperante.
MK _ XBUTTON1 0x0020	O primeiro botão X está inoperante.

VALOR	SIGNIFICADO
MK _ XBUTTON2 0x0040	O segundo botão X está inoperante.

lParam

A palavra de ordem inferior Especifica a coordenada x do ponteiro, em relação ao canto superior esquerdo da tela.

A palavra de ordem superior especifica a coordenada y do ponteiro, em relação ao canto superior esquerdo da tela.

Retornar valor

Se um aplicativo processar essa mensagem, ele deverá retornar zero.

Comentários

Use o código a seguir para obter as informações no parâmetro *wParam* :

```
fwKeys = GET_KEYSTATE_WPARAM(wParam);
zDelta = GET_WHEEL_DELTA_WPARAM(wParam);
```

Use o código a seguir para obter a posição horizontal e vertical:

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

Conforme observado acima, a coordenada x está **na ordem inferior** do valor de retorno; a coordenada y está no **intervalo** de ordem superior (ambos representam valores *assinados* porque podem receber valores negativos em sistemas com vários monitores). Se o valor de retorno for atribuído a uma variável, você poderá usar a macro MAKEPOINTS para obter uma estrutura de **pontos** do valor de retorno. Você também pode usar a macro Get _ X _ lParam ou Get _ y _ lParam para extrair a coordenada x ou Y.

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor, pois essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades não assinadas.

A rotação da roda será um múltiplo do **Wheel _ Delta**, que é definido em 120. Esse é o limite para a ação a ser tomada e uma ação desse tipo (por exemplo, rolagem de um incremento) deve ocorrer para cada Delta.

O Delta foi definido como 120 para permitir que a Microsoft ou outros fornecedores criem rodas de resolução mais fina (uma roda de rotação livre sem entalhes) para enviar mais mensagens por rotação, mas com um valor menor em cada mensagem. Para usar esse recurso, você pode adicionar os valores Delta de entrada até que o **Wheel _ Delta** seja atingido (para que uma rotação Delta obtenha a mesma resposta) ou role as linhas parciais em resposta às mensagens mais frequentes. Você também pode escolher a granularidade da rolagem e acumular deltas até que ele seja atingido.

Observe que não há nenhum *fwKeys* para **MSH _ MOUSEWHEEL**. Caso contrário, os parâmetros são

exatamente iguais aos do **WM _ MOUSEWHEEL**.

Cabe ao aplicativo encaminhar **MSH _ MOUSEWHEEL** a quaisquer objetos ou controles inseridos. O aplicativo é necessário para enviar a mensagem a um aplicativo OLE incorporado ativo. É opcional que o aplicativo o envia para um controle habilitado para roda com foco. Se o aplicativo enviar a mensagem a um controle, ele poderá verificar o valor de retorno para ver se a mensagem foi processada. Os controles são necessários para retornar um valor **true** se eles processarem a mensagem.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	WinUser. h (incluir Windowsx. h)

Confira também

Referência

OBTER _ _ wParam KeyState

OBTER _ X _ lParam

OBTER _ _ lParam Y

OBTER _ _ wParam Delta do volante _

HIWORD

LOWORD

evento do mouse _

Conceitual

Entrada do mouse

Outros recursos

GetSystemMetrics

MAKEPOINTS

FAIXAS

SystemParametersInfo

Mensagem WM _ NCHITTEST

15/04/2022 • 3 minutes to read

Enviado para uma janela para determinar qual parte da janela corresponde a uma coordenada de tela específica. Isso pode acontecer, por exemplo, quando o cursor se move, quando um botão do mouse é pressionado ou liberado ou em resposta a uma chamada para uma função como **WindowFromPoint**. Se o mouse não for capturado, a mensagem será enviada para a janela abaixo do cursor. Caso contrário, a mensagem será enviada para a janela que capturou o mouse.

Uma janela recebe essa mensagem por meio de **sua função WindowProc.**

```
#define WM_NCHITTEST                    0x0084
```

Parâmetros

wParam

Este parâmetro não é usado.

lParam

A palavra de ordem baixa especifica a coordenada X do cursor. A coordenada é relativa ao canto superior esquerdo da tela.

A palavra de ordem alta especifica a coordenada y do cursor. A coordenada é relativa ao canto superior esquerdo da tela.

Retornar valor

O valor de retorno **da função DefWindowProc** é um dos valores a seguir, indicando a posição do ponto de acionamento do cursor.

VALOR/CÓDIGO DE RETORNO	DESCRIÇÃO
HTBORDER 18	Na borda de uma janela que não tem uma borda deizing.
HTBOTTOM 15	Na borda inferior horizontal de uma janela reizável (o usuário pode clicar no mouse para reessar a janela verticalmente).
HTBOTTOMLEFT 16	No canto inferior esquerdo de uma borda de uma janela reizável (o usuário pode clicar no mouse para ressarciá-lo diagonalmente).
HTBOTTOMRIGHT 17	No canto inferior direito de uma borda de uma janela reizável (o usuário pode clicar no mouse para ressarciá-lo diagonalmente).

VALOR/CÓDIGO DE RETORNO	DESCRIÇÃO
HTCAPTION 2	Em uma barra de título.
HTCLIENT 1	Em uma área de cliente.
HTCLOSE 20	Em um **botão** Fechar.
HTERROR -2	Na tela de fundo ou em uma linha divisória entre janelas (o mesmo que **HTNOWHERE**, exceto que a **função DefWindowProc** produz um aviso de aviso do sistema para indicar um erro).
HTGROWBOX 4	Em uma caixa de tamanho (o mesmo que **HTSIZE**).
HTHELP 21	Em um **botão ajuda.**
HTHSCROLL 6	Em uma barra de rolagem horizontal.
HTLEFT 10	Na borda esquerda de uma janela reizável (o usuário pode clicar no mouse para reessar a janela horizontalmente).
HTMENU 5	Em um menu.
HTMAXBUTTON 9	Em um **botão Maximizar.**
HTMINBUTTON 8	Em um **botão** Minimizar.
HTNOWHERE 0	Na tela de fundo ou em uma linha de divisão entre janelas.

VALOR/CÓDIGO DE RETORNO	DESCRIÇÃO
HTREDUCE 8	Em um **botão** Minimizar.
HTRIGHT 11	Na borda direita de uma janela reizável (o usuário pode clicar no mouse para reessar a janela horizontalmente).
HTSIZE 4	Em uma caixa de tamanho (igual a **HTGROWBOX**).
HTSYSMENU 3	Em um menu de janela ou em **um botão** Fechar em uma janela filho.
HTTOP 12	Na borda superior horizontal de uma janela.
HTTOPLEFT 13	No canto superior esquerdo de uma borda da janela.
HTTOPRIGHT 14	No canto superior direito de uma borda da janela.
HTTRANSPARENT -1	Em uma janela atualmente coberta por outra janela no mesmo thread (a mensagem será enviada para janelas subjacentes no mesmo thread até que uma delas retorne um código que não seja **HTTRANSPARENT**).
HTVSCROLL 7	Na barra de rolagem vertical.
HTZOOM 9	Em um **botão Maximizar.**

Comentários

Use o código a seguir para obter a posição horizontal e vertical:

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

Conforme mencionado acima, a coordenada x está na ordem baixa **a menos** que o valor de retorno; a coordenada y está em ordem alta curta **(ambos** representam valores assinados porque podem receber valores negativos em sistemas com vários monitores). Se o valor de retorno for atribuído a uma variável, você poderá

usar a macro MAKEPOINTS para obter uma estrutura POINTS do valor de retorno. Você também pode usar a macro GET _ X _ LPARAM ou GET Y _ _ LPARAM para extrair a coordenada x ou y.

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor porque essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas, e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades sem sinal.

Windows Vista: Ao criar quadros personalizados que incluem os botões de legenda padrão, essa mensagem deve primeiro ser passada para a **função DwmDefWindowProc.** Isso permite que o Gerenciador de Janelas da Área de Trabalho (DWM) forneça testes de clique para os botões de legendas. Se **DwmDefWindowProc** não tratar a mensagem, poderá ser necessário um processamento posterior do **WM _ NCHITTEST.**

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (inclua Windowsx.h)

Confira também

Referência

Defwindowproc

GET _ X _ LPARAM

GET _ Y _ LPARAM

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

FAIXAS

Mensagem do WM _ NCLBUTTONDBLCLK

15/04/2022 • 2 minutes to read

Postado quando o usuário clica duas vezes com o botão esquerdo do mouse enquanto o cursor está dentro da área não cliente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.

Uma janela recebe essa mensagem por meio de sua função **WindowProc** .

```
#define WM_NCLBUTTONDBLCLK              0x00A3
```

Parâmetros

wParam

O valor de teste de clique retornado pela função **DefWindowProc** como resultado do processamento da mensagem **_ NCHITTEST do WM** . Para obter uma lista de valores de teste de clique, consulte **WM _ NCHITTEST**.

lParam

Uma estrutura de **pontos** que contém as coordenadas x e y do cursor. As coordenadas são relativas ao canto superior esquerdo da tela.

Retornar valor

Se um aplicativo processar essa mensagem, ele deverá retornar zero.

Comentários

Você também pode usar as macros **Get _ X _ lParam** e **Get _ y _ lParam** para extrair os valores das coordenadas X e y do *lParam*.

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor, pois essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades não assinadas.

Por padrão, a função **DefWindowProc** testa o ponto especificado para descobrir o local do cursor e executa a ação apropriada. Se apropriado, **DefWindowProc** envia a mensagem do **WM _ SYSCOMMAND** para a janela.

Uma janela não precisa ter o estilo **cs _ DBLCLKS** para receber mensagens do **WM _ NCLBUTTONDBLCLK** .

O sistema gera uma mensagem do **WM _ NCLBUTTONDBLCLK** quando o usuário pressiona, libera e novamente pressiona o botão esquerdo do mouse dentro do limite de tempo de clique duplo do sistema. Clicar duas vezes no botão esquerdo do mouse gera quatro mensagens: **WM _ NCLBUTTONDOWN**, **WM _**

NCLBUTTONUP, **WM _ NCLBUTTONDBLCLK** e **WM _ NCLBUTTONUP** novamente.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	WinUser. h (incluir Windowsx. h)

Confira também

Referência

DefWindowProc

OBTER _ X _ lParam

OBTER _ _ lParam Y

NCHITTEST do WM _

NCLBUTTONDOWN do WM _

NCLBUTTONUP do WM _

SYSCOMMAND do WM _

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

FAIXAS

Mensagem _ WM NCLBUTTONDOWN

15/04/2022 • 2 minutes to read

Postado quando o usuário pressiona o botão esquerdo do mouse enquanto o cursor está dentro da área não dependente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.

Uma janela recebe essa mensagem por meio de **sua função WindowProc.**

```
#define WM_NCLBUTTONDOWN                0x00A1
```

Parâmetros

wParam

O valor de teste de acerto retornado pela **função DefWindowProc** como resultado do processamento da mensagem **WM _ NCHITTEST.** Para ver uma lista de valores de teste de acerto, consulte **WM _ NCHITTEST**.

lParam

Uma **estrutura POINTS** que contém as coordenadas x e y do cursor. As coordenadas são relativas ao canto superior esquerdo da tela.

Retornar valor

Se um aplicativo processa essa mensagem, ele deve retornar zero.

Comentários

A **função DefWindowProc** testa o ponto especificado para localizar o local do cursor e executa a ação apropriada. Se apropriado, **DefWindowProc** envia a mensagem **WM _ SYSCOMMAND** para a janela.

Você também pode usar as **macros GET _ X _ LPARAM** e **GET Y _ _ LPARAM** para extrair os valores das coordenadas x e y de *lParam*.

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

IMPORTANT

Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor porque essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas, e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades sem sinal.

Requisitos

REQUISITO	VALOR

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (inclua Windowsx.h)

Confira também

Referência

Defwindowproc

GET _ X _ LPARAM

GET _ Y _ LPARAM

WM _ NCHITTEST

WM _ NCLBUTTONDBLCLK

WM _ NCLBUTTONUP

WM _ SYSCOMMAND

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

PONTOS

Mensagem do WM _ NCLBUTTONUP

15/04/2022 • 2 minutes to read

Postado quando o usuário libera o botão esquerdo do mouse enquanto o cursor está dentro da área não cliente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.

Uma janela recebe essa mensagem por meio de sua função **WindowProc** .

```
#define WM_NCLBUTTONUP                  0x00A2
```

Parâmetros

wParam

O valor de teste de clique retornado pela função **DefWindowProc** como resultado do processamento da mensagem **_ NCHITTEST do WM** . Para obter uma lista de valores de teste de clique, consulte **WM _ NCHITTEST**.

lParam

Uma estrutura de **pontos** que contém as coordenadas x e y do cursor. As coordenadas são relativas ao canto superior esquerdo da tela.

Retornar valor

Se um aplicativo processar essa mensagem, ele deverá retornar zero.

Comentários

A função **DefWindowProc** testa o ponto especificado para descobrir o local do cursor e executa a ação apropriada. Se apropriado, **DefWindowProc** envia a mensagem do **WM _ SYSCOMMAND** para a janela.

Você também pode usar as macros **Get _ X _ lParam** e **Get _ y _ lParam** para extrair os valores das coordenadas X e y do *lParam*.

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

IMPORTANT

Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor, pois essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades não assinadas.

Se for apropriado fazer isso, o sistema enviará a mensagem **_ SYSCOMMAND do WM** para a janela.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	WinUser. h (incluir Windowsx. h)

Confira também

Referência

DefWindowProc

OBTER _ X _ lParam

OBTER _ _ lParam Y

NCHITTEST do WM _

NCLBUTTONDBLCLK do WM _

NCLBUTTONDOWN do WM _

SYSCOMMAND do WM _

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

FAIXAS

Mensagem do WM _ NCMBUTTONDBLCLK

15/04/2022 • 2 minutes to read

Postado quando o usuário clica duas vezes no botão do meio do mouse enquanto o cursor está dentro da área não cliente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.

Uma janela recebe essa mensagem por meio de sua função **WindowProc** .

```
#define WM_NCMBUTTONDBLCLK              0x00A9
```

Parâmetros

wParam

O valor de teste de clique retornado pela função **DefWindowProc** como resultado do processamento da mensagem **_ NCHITTEST do WM** . Para obter uma lista de valores de teste de clique, consulte **WM _ NCHITTEST**.

lParam

Uma estrutura de **pontos** que contém as coordenadas x e y do cursor. As coordenadas são relativas ao canto superior esquerdo da tela.

Retornar valor

Se um aplicativo processar essa mensagem, ele deverá retornar zero.

Comentários

Uma janela não precisa ter o estilo **cs _ DBLCLKS** para receber mensagens do **WM _ NCMBUTTONDBLCLK** .

O sistema gera uma mensagem do **WM _ NCMBUTTONDBLCLK** quando o usuário pressiona, libera e novamente pressiona o botão do meio do mouse dentro do limite de tempo de clique duplo do sistema. Clicar duas vezes no botão do meio do mouse gera quatro mensagens : **_ WM NCMBUTTONDOWN**, **WM _ NCMBUTTONUP**, **WM _ NCMBUTTONDBLCLK** e **WM _ NCMBUTTONUP** novamente.

Você também pode usar as macros **Get _ X _ lParam** e **Get _ y _ lParam** para extrair os valores das coordenadas X e y do *lParam*.

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor, pois essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades não assinadas.

Se for apropriado fazer isso, o sistema enviará a mensagem **_ SYSCOMMAND do WM** para a janela.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	WinUser. h (incluir Windowsx. h)

Confira também

Referência

DefWindowProc

OBTER _ X _ lParam

OBTER _ _ lParam Y

NCHITTEST do WM _

NCMBUTTONDOWN do WM _

NCMBUTTONUP do WM _

SYSCOMMAND do WM _

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

FAIXAS

Mensagem _ WM NCMBUTTONDOWN

15/04/2022 • 2 minutes to read

Postado quando o usuário pressiona o botão do meio do mouse enquanto o cursor está dentro da área não dependente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.

Uma janela recebe essa mensagem por meio de **sua função WindowProc.**

```
#define WM_NCMBUTTONDOWN                0x00A7
```

Parâmetros

wParam

O valor de teste de acerto retornado pela **função DefWindowProc** como resultado do processamento da mensagem **WM _ NCHITTEST.** Para ver uma lista de valores de teste de acerto, consulte **WM _ NCHITTEST**.

lParam

Uma **estrutura POINTS** que contém as coordenadas x e y do cursor. As coordenadas são relativas ao canto superior esquerdo da tela.

Retornar valor

Se um aplicativo processa essa mensagem, ele deve retornar zero.

Comentários

Você também pode usar as **macros GET _ X _ LPARAM** e **GET Y _ _ LPARAM** para extrair os valores das coordenadas x e y de *lParam*.

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor porque essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas, e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades sem sinal.

Se for apropriado fazer isso, o sistema enviará a mensagem **WM _ SYSCOMMAND** para a janela.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]

REQUISITO	VALOR
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (inclua Windowsx.h)

Confira também

Referência

Defwindowproc

GET _ X _ LPARAM

GET _ Y _ LPARAM

WM _ NCHITTEST

WM _ NCMBUTTONDBLCLK

WM _ NCMBUTTONUP

WM _ SYSCOMMAND

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

PONTOS

Mensagem do WM _ NCMBUTTONUP

15/04/2022 • 2 minutes to read

Postado quando o usuário libera o botão do meio do mouse enquanto o cursor está dentro da área não cliente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.

Uma janela recebe essa mensagem por meio de sua função **WindowProc** .

```
#define WM_NCMBUTTONUP                  0x00A8
```

Parâmetros

wParam

O valor de teste de clique retornado pela função **DefWindowProc** como resultado do processamento da mensagem **_ NCHITTEST do WM** . Para obter uma lista de valores de teste de clique, consulte **WM _ NCHITTEST**.

lParam

Uma estrutura de **pontos** que contém as coordenadas x e y do cursor. As coordenadas são relativas ao canto superior esquerdo da tela.

Retornar valor

Se um aplicativo processar essa mensagem, ele deverá retornar zero.

Comentários

Você também pode usar as macros **Get _ X _ lParam** e **Get _ y _ lParam** para extrair os valores das coordenadas X e y do *lParam*.

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor, pois essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades não assinadas.

Se for apropriado fazer isso, o sistema enviará a mensagem **_ SYSCOMMAND do WM** para a janela.

Requisitos

REQUISITO	VALOR

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	WinUser. h (incluir Windowsx. h)

Confira também

Referência

DefWindowProc

OBTER _ X _ lParam

OBTER _ _ lParam Y

NCHITTEST do WM _

NCMBUTTONDBLCLK do WM _

NCMBUTTONDOWN do WM _

SYSCOMMAND do WM _

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

FAIXAS

Mensagem _ WM NCMOUSEHOVER

15/04/2022 • 2 minutes to read

Postado em uma janela quando o cursor passar o mouse sobre a área não dependente da janela pelo período de tempo especificado em uma chamada anterior para **TrackMouseEvent**.

Uma janela recebe essa mensagem por meio de **sua função WindowProc.**

```
#define WM_NCMOUSEHOVER                 0x02A0
```

Parâmetros

wParam

O valor de teste de acerto retornado pela **função DefWindowProc** como resultado do processamento da mensagem **WM _ NCHITTEST.** Para ver uma lista de valores de teste de acerto, consulte **WM _ NCHITTEST**.

lParam

Uma **estrutura POINTS** que contém as coordenadas x e y do cursor. As coordenadas são relativas ao canto superior esquerdo da tela.

Retornar valor

Se um aplicativo processa essa mensagem, ele deve retornar zero.

Comentários

O acompanhamento de foco é interrompido quando essa mensagem é gerada. O aplicativo deverá chamar **TrackMouseEvent** novamente se exigir um acompanhamento posterior do comportamento de foco do mouse.

Você também pode usar as **macros GET _ X _ LPARAM** e **GET Y _ _ LPARAM** para extrair os valores das coordenadas x e y de *lParam*.

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

IMPORTANT

Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor porque essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas, e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades sem sinal.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]

REQUISITO	VALOR
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (inclua Windowsx.h)

Confira também

Referência

Defwindowproc

GET _ X _ LPARAM

GET _ Y _ LPARAM

Trackmouseevent

TRACKMOUSEEVENT

WM _ NCHITTEST

WM _ MOUSEHOVER

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

PONTOS

Mensagem do WM _ NCMOUSELEAVE

15/04/2022 • 2 minutes to read

Postado em uma janela quando o cursor sai da área não cliente da janela especificada em uma chamada anterior para **TrackMouseEvent**.

Uma janela recebe essa mensagem por meio de sua função **WindowProc** .

```
#define WM_NCMOUSELEAVE                 0x02A2
```

Parâmetros

wParam

Esse parâmetro não é usado e deve ser zero.

lParam

Esse parâmetro não é usado e deve ser zero.

Retornar valor

Se um aplicativo processar essa mensagem, ele deverá retornar zero.

Comentários

Todo o controle solicitado por **TrackMouseEvent** é cancelado quando esta mensagem é gerada. O aplicativo deve chamar **TrackMouseEvent** quando o mouse reinserir sua janela se precisar de mais controle sobre o comportamento de focalização do mouse.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser. h (incluir Windows. h)

Confira também

Referência

TrackMouseEvent

TRACKMOUSEEVENT

SYSCOMMAND do WM _

MOUSELEAVE do WM _

Conceitual

Entrada do mouse

Mensagem do WM _ NCMOUSEMOVE

15/04/2022 • 2 minutes to read

Postado em uma janela quando o cursor é movido dentro da área não cliente da janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.

Uma janela recebe essa mensagem por meio de sua função **WindowProc** .

```
#define WM_NCMOUSEMOVE                  0x00A0
```

Parâmetros

wParam

O valor de teste de clique retornado pela função **DefWindowProc** como resultado do processamento da mensagem **_ NCHITTEST do WM** . Para obter uma lista de valores de teste de clique, consulte **WM _ NCHITTEST**.

lParam

Uma estrutura de **pontos** que contém as coordenadas x e y do cursor. As coordenadas são relativas ao canto superior esquerdo da tela.

Retornar valor

Se um aplicativo processar essa mensagem, ele deverá retornar zero.

Comentários

Se for apropriado fazer isso, o sistema enviará a mensagem **_ SYSCOMMAND do WM** para a janela.

Você também pode usar as macros **Get _ X _ lParam** e **Get _ y _ lParam** para extrair os valores das coordenadas X e y do *lParam*.

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

IMPORTANT

Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor, pois essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades não assinadas.

Requisitos

REQUISITO	VALOR

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	WinUser. h (incluir Windowsx. h)

Confira também

Referência

DefWindowProc

OBTER _ X _ lParam

OBTER _ _ lParam Y

NCHITTEST do WM _

SYSCOMMAND do WM _

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

FAIXAS

Mensagem _ WM NCRBUTTONDBLCLK

15/04/2022 • 2 minutes to read

Postado quando o usuário clica duas vezes no botão direito do mouse enquanto o cursor está dentro da área não dependente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.

Uma janela recebe essa mensagem por meio de **sua função WindowProc.**

```
#define WM_NCRBUTTONDBLCLK              0x00A6
```

Parâmetros

wParam

O valor de teste de acerto retornado pela **função DefWindowProc** como resultado do processamento da mensagem **WM _ NCHITTEST.** Para ver uma lista de valores de teste de acerto, consulte **WM _ NCHITTEST**.

lParam

Uma **estrutura POINTS** que contém as coordenadas x e y do cursor. As coordenadas são relativas ao canto superior esquerdo da tela.

Retornar valor

Se um aplicativo processa essa mensagem, ele deve retornar zero.

Comentários

Uma janela não precisa ter o **estilo _ DBLCLKS CS** para receber **mensagens WM _ NCRBUTTONDBLCLK.**

O sistema gera uma mensagem **WM _ NCRBUTTONDBLCLK** quando o usuário pressiona, libera e pressiona novamente o botão direito do mouse dentro do limite de tempo de clique duplo do sistema. Clicar duas vezes no botão direito do mouse gera quatro mensagens: **WM _ NCRBUTTONDOWN, WM _ NCRBUTTONUP, WM _ NCRBUTTONDBLCLK** e **WM _ NCRBUTTONUP** novamente.

Você também pode usar as **macros GET _ X _ LPARAM** e **GET Y _ _ LPARAM** para extrair os valores das coordenadas x e y de *lParam*.

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor porque essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas, e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades sem sinal.

Se for apropriado fazer isso, o sistema enviará a mensagem **WM _ SYSCOMMAND** para a janela.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (inclua Windowsx.h)

Confira também

Referência

Defwindowproc

GET _ X _ LPARAM

GET _ Y _ LPARAM

WM _ NCHITTEST

WM _ NCRBUTTONDOWN

WM _ NCRBUTTONUP

WM _ SYSCOMMAND

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

PONTOS

Mensagem WM _ NCRBUTTONDOWN

15/04/2022 • 2 minutes to read

Postado quando o usuário pressiona o botão direito do mouse enquanto o cursor está dentro da área não dependente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.

Uma janela recebe essa mensagem por meio de **sua função WindowProc.**

```
#define WM_NCRBUTTONDOWN                0x00A4
```

Parâmetros

wParam

O valor de teste de acerto retornado pela **função DefWindowProc** como resultado do processamento da mensagem **WM _ NCHITTEST.** Para ver uma lista de valores de teste de acerto, consulte **WM _ NCHITTEST**.

lParam

Uma **estrutura POINTS** que contém as coordenadas x e y do cursor. As coordenadas são relativas ao canto superior esquerdo da tela.

Retornar valor

Se um aplicativo processa essa mensagem, ele deve retornar zero.

Comentários

Você também pode usar as **macros GET _ X _ LPARAM** e **GET Y _ _ LPARAM** para extrair os valores das coordenadas x e y de *lParam*.

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor porque essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas, e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades sem sinal.

Se for apropriado fazer isso, o sistema enviará a mensagem **WM _ SYSCOMMAND** para a janela.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]

REQUISITO	VALOR
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (inclua Windowsx.h)

Confira também

Referência

Defwindowproc

GET _ X _ LPARAM

GET _ Y _ LPARAM

WM _ NCHITTEST

WM _ NCRBUTTONDBLCLK

WM _ NCRBUTTONUP

WM _ SYSCOMMAND

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

PONTOS

Mensagem do WM _ NCRBUTTONUP

15/04/2022 • 2 minutes to read

Postado quando o usuário libera o botão direito do mouse enquanto o cursor está dentro da área não cliente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.

Uma janela recebe essa mensagem por meio de sua função **WindowProc** .

```
#define WM_NCRBUTTONUP                  0x00A5
```

Parâmetros

wParam

O valor de teste de clique retornado pela função **DefWindowProc** como resultado do processamento da mensagem **_ NCHITTEST do WM** . Para obter uma lista de valores de teste de clique, consulte **WM _ NCHITTEST**.

lParam

Uma estrutura de **pontos** que contém as coordenadas x e y do cursor. As coordenadas são relativas ao canto superior esquerdo da tela.

Retornar valor

Se um aplicativo processar essa mensagem, ele deverá retornar zero.

Comentários

Você também pode usar as macros **Get _ X _ lParam** e **Get _ y _ lParam** para extrair os valores das coordenadas X e y do *lParam*.

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor, pois essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades não assinadas.

Se for apropriado fazer isso, o sistema enviará a mensagem **_ SYSCOMMAND do WM** para a janela.

Requisitos

REQUISITO	VALOR

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	WinUser. h (incluir Windowsx. h)

Confira também

Referência

DefWindowProc

OBTER _ X _ lParam

OBTER _ _ lParam Y

NCHITTEST do WM _

NCRBUTTONDBLCLK do WM _

NCRBUTTONDOWN do WM _

SYSCOMMAND do WM _

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

FAIXAS

Mensagem do WM _ NCXBUTTONDBLCLK

15/04/2022 • 2 minutes to read

Postado quando o usuário clica duas vezes no primeiro ou no segundo botão X enquanto o cursor está na área não cliente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem não será postada.

Uma janela recebe essa mensagem por meio de sua função **WindowProc** .

```
#define WM_NCXBUTTONDBLCLK              0x00AD
```

Parâmetros

wParam

A palavra de ordem inferior Especifica o valor de teste de clique retornado pela função **DefWindowProc** do processamento da mensagem do **WM _ NCHITTEST** . Para obter uma lista de valores de teste de clique, consulte **WM _ NCHITTEST**.

A palavra de ordem superior indica qual botão foi clicado duas vezes. Pode ser um dos seguintes valores.

VALOR	SIGNIFICADO
XButton1 0x0001	O primeiro botão X foi clicado duas vezes.
XButton2 0x0002	O segundo botão X foi clicado duas vezes.

lParam

Um ponteiro para uma estrutura de **pontos** que contém as coordenadas x e y do cursor. As coordenadas são relativas ao canto superior esquerdo da tela.

Retornar valor

Se um aplicativo processar essa mensagem, ele deverá retornar **true**. Para obter mais informações sobre como processar o valor de retorno, consulte a seção comentários.

Comentários

Use o código a seguir para obter as informações no parâmetro *wParam* .

```
nHittest = GET_NCHITTEST_WPARAM(wParam);
fwButton = GET_XBUTTON_WPARAM(wParam);
```

Você também pode usar o código a seguir para obter as coordenadas x e y do *lParam*:

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

IMPORTANT

Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor, pois essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades não assinadas.

Por padrão, a função **DefWindowProc** testa o ponto especificado para obter a posição do cursor e executa a ação apropriada. Se apropriado, ele envia a **mensagem _ SYSCOMMAND do WM** para a janela.

Uma janela não precisa ter o estilo **cs _ DBLCLKS** para receber mensagens do **WM _ NCXBUTTONDBLCLK** . O sistema gera uma mensagem do **WM _ NCXBUTTONDBLCLK** quando o usuário pressiona, libera e novamente pressiona um botão X dentro do limite de tempo de clique duplo do sistema. Clicar duas vezes em um desses botões gera quatro mensagens: **WM _ NCXBUTTONDOWN**, **WM _ NCXBUTTONUP**, **WM _ NCXBUTTONDBLCLK** e **WM _ NCXBUTTONUP** novamente.

Ao contrário das mensagens do **WM _ NCLBUTTONDBLCLK**, do **WM _ NCMBUTTONDBLCLK**e do **WM _ NCRBUTTONDBLCLK** , um aplicativo deve retornar **true** dessa mensagem se a processar. isso permitirá que o software que simula essa mensagem em sistemas Windows anteriores a Windows 2000 para determinar se o procedimento de janela processou a mensagem ou se chamou **DefWindowProc** para processá-la.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	WinUser. h (incluir Windowsx. h)

Confira também

Referência

DefWindowProc

OBTER _ X _ lParam

OBTER _ _ lParam Y

NCHITTEST do WM _

NCXBUTTONDOWN do WM _

NCXBUTTONUP do WM _

SYSCOMMAND do WM _

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

FAIXAS

Mensagem do WM _ NCXBUTTONDOWN

15/04/2022 • 2 minutes to read

Postado quando o usuário pressiona o primeiro ou o segundo botão X enquanto o cursor está na área não cliente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem *não* será postada.

Uma janela recebe essa mensagem por meio de sua função **WindowProc** .

```
#define WM_NCXBUTTONDOWN                0x00AB
```

Parâmetros

wParam

A palavra de ordem inferior Especifica o valor de teste de clique retornado pela função **DefWindowProc** do processamento da mensagem do **WM _ NCHITTEST** . Para obter uma lista de valores de teste de clique, consulte **WM _ NCHITTEST**. A palavra de ordem superior indica qual botão foi pressionado. Pode ser um dos seguintes valores.

VALOR	SIGNIFICADO
XButton1 0x0001	O primeiro botão X foi pressionado.
XButton2 0x0002	O segundo botão X foi pressionado.

lParam

Um ponteiro para uma estrutura de **pontos** que contém as coordenadas x e y do cursor. As coordenadas são relativas ao canto superior esquerdo da tela.

Retornar valor

Se um aplicativo processar essa mensagem, ele deverá retornar **true**. Para obter mais informações sobre como processar o valor de retorno, consulte a seção comentários.

Comentários

Use o código a seguir para obter as informações no parâmetro *wParam* .

```
nHittest = GET_NCHITTEST_WPARAM(wParam);
fwButton = GET_XBUTTON_WPARAM(wParam);
```

Você também pode usar o código a seguir para obter as coordenadas x e y do *lParam*:

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor, pois essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades não assinadas.

Por padrão, a função **DefWindowProc** testa o ponto especificado para obter a posição do cursor e executa a ação apropriada. Se apropriado, ele envia a **mensagem _ SYSCOMMAND do WM** para a janela.

Ao contrário das mensagens do **WM _ NCLBUTTONDOWN**, do **WM _ NCMBUTTONDOWN** e do **WM _ NCRBUTTONDOWN** , um aplicativo deve retornar **true** dessa mensagem se a processar. isso permitirá que o software que simula essa mensagem em sistemas Windows anteriores a Windows 2000 para determinar se o procedimento de janela processou a mensagem ou se chamou **DefWindowProc** para processá-la.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	WinUser. h (incluir Windowsx. h)

Confira também

Referência

DefWindowProc

OBTER _ X _ lParam

OBTER _ _ lParam Y

NCHITTEST do WM _

NCXBUTTONDBLCLK do WM _

NCXBUTTONUP do WM _

SYSCOMMAND do WM _

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

Mensagem _ WM NCXBUTTONUP

15/04/2022 • 2 minutes to read

Postado quando o usuário libera o primeiro ou segundo botão X enquanto o cursor está na área não dependente de uma janela. Essa mensagem é postada na janela que contém o cursor. Se uma janela tiver capturado o mouse, essa mensagem *não será* postada.

Uma janela recebe essa mensagem por meio de **sua função WindowProc.**

```
#define WM_NCXBUTTONUP                  0x00AC
```

Parâmetros

wParam

A palavra de ordem baixa especifica o valor de teste de acerto retornado pela função **DefWindowProc** do processamento da mensagem **WM _ NCHITTEST.** Para ver uma lista de valores de teste de acerto, consulte **WM _ NCHITTEST**.

A palavra de ordem alta indica qual botão foi liberado. Pode ser um dos seguintes valores.

VALOR	SIGNIFICADO
XBUTTON1 0x0001	O primeiro botão X foi liberado.
XBUTTON2 0x0002	O segundo botão X foi liberado.

lParam

Um ponteiro para uma **estrutura POINTS** que contém as coordenadas x e y do cursor. As coordenadas são relativas ao canto superior esquerdo da tela.

Retornar valor

Se um aplicativo processa essa mensagem, ele deve retornar **TRUE.** Para obter mais informações sobre como processar o valor de retorno, consulte a seção Comentários.

Comentários

Use o código a seguir para obter as informações no *parâmetro wParam.*

```
nHittest = GET_NCHITTEST_WPARAM(wParam);
fwButton = GET_XBUTTON_WPARAM(wParam);
```

Você também pode usar o código a seguir para obter as coordenadas x e y de *lParam*:

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor porque essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas, e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades sem sinal.

Por padrão, a **função DefWindowProc** testa o ponto especificado para obter a posição do cursor e executa a ação apropriada. Se apropriado, ele envia a **_ mensagem WM SYSCOMMAND** para a janela.

Ao contrário **das mensagens WM _ NCLBUTTONUP, WM _ NCMBUTTONUP** e **WM _ NCRBUTTONUP,** um aplicativo deverá retornar **TRUE** dessa mensagem se processá-lo. Isso permitirá que o software que simula essa mensagem em sistemas Windows anteriores ao Windows 2000 determine se o procedimento de janela processou a mensagem ou chamou **DefWindowProc** para processá-la.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (inclua Windowsx.h)

Confira também

Referência

Defwindowproc

GET _ X _ LPARAM

GET _ Y _ LPARAM

WM _ NCHITTEST

WM _ NCXBUTTONDBLCLK

WM _ NCXBUTTONDOWN

WM _ SYSCOMMAND

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

Mensagem WM _ RBUTTONDBLCLK

15/04/2022 • 2 minutes to read

Postado quando o usuário clica duas vezes no botão direito do mouse enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.

Uma janela recebe essa mensagem por meio de **sua função WindowProc.**

```
#define WM_RBUTTONDBLCLK                0x0206
```

Parâmetros

wParam

Indica se várias chaves virtuais estão inoadas. Esse parâmetro pode ser um ou mais dos valores a seguir.

VALOR	SIGNIFICADO
MK _ CONTROLE 0x0008	A tecla CTRL está inocizada.
MK _ LBUTTON 0x0001	O botão esquerdo do mouse está ino mouse.
MK _ MBUTTON 0x0010	O botão do meio do mouse está ino mouse.
MK _ RBUTTON 0x0002	O botão direito do mouse está ino mouse.
MK _ SHIFT 0x0004	A tecla SHIFT está inobada.
MK _ XBUTTON1 0x0020	O primeiro botão X está ino mouse.
MK _ XBUTTON2 0x0040	O segundo botão X está ino mouse.

lParam

A palavra de ordem baixa especifica a coordenada X do cursor. A coordenada é relativa ao canto superior

esquerdo da área do cliente.

A palavra de ordem alta especifica a coordenada y do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

Retornar valor

Se um aplicativo processa essa mensagem, ele deve retornar zero.

Comentários

Somente as janelas que têm o estilo _ **DBLCLKS** do CS podem receber mensagens **WM _ RBUTTONDBLCLK,** que o sistema gera sempre que o usuário pressiona, libera e pressiona novamente o botão direito do mouse dentro do limite de tempo de clique duplo do sistema. Clicar duas vezes no botão direito do mouse gera quatro mensagens: WM _ RBUTTONDOWN, WM _ RBUTTONUP, **WM _ RBUTTONDBLCLK** e **WM _ RBUTTONUP** novamente.

Use o código a seguir para obter a posição horizontal e vertical:

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

Conforme mencionado acima, a coordenada x está na ordem baixa **a menos** que o valor de retorno; a coordenada y está em ordem alta curta **(ambos** representam valores assinados porque podem receber valores negativos em sistemas com vários monitores). Se o valor de retorno for atribuído a uma variável, você poderá usar a macro MAKEPOINTS para obter uma estrutura POINTS do valor de retorno. Você também pode usar a macro GET _ X _ LPARAM ou GET Y _ _ LPARAM para extrair a coordenada x ou y.

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor porque essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas, e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades sem sinal.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (inclua Windowsx.h)

Confira também

Referência

GET _ X _ LPARAM

GET _ Y _ LPARAM

GetCapture

GetDoubleClickTime

Setcapture

SetDoubleClickTime

WM _ RBUTTONDOWN

WM _ RBUTTONUP

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

PONTOS

Mensagem WM _ RBUTTONDOWN

15/04/2022 • 2 minutes to read

Postado quando o usuário pressiona o botão direito do mouse enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.

Uma janela recebe essa mensagem por meio de **sua função WindowProc.**

```
#define WM_RBUTTONDOWN                  0x0204
```

Parâmetros

wParam

Indica se várias chaves virtuais estão inoadas. Esse parâmetro pode ser um ou mais dos valores a seguir.

VALOR	SIGNIFICADO
MK _ CONTROLE 0x0008	A tecla CTRL está inocizada.
MK _ LBUTTON 0x0001	O botão esquerdo do mouse está ino mouse.
MK _ MBUTTON 0x0010	O botão do meio do mouse está ino mouse.
MK _ RBUTTON 0x0002	O botão direito do mouse está ino mouse.
MK _ SHIFT 0x0004	A tecla SHIFT está inobada.
MK _ XBUTTON1 0x0020	O primeiro botão X está ino mouse.
MK _ XBUTTON2 0x0040	O segundo botão X está ino mouse.

lParam

A palavra de ordem baixa especifica a coordenada X do cursor. A coordenada é relativa ao canto superior

esquerdo da área do cliente.

A palavra de ordem alta especifica a coordenada y do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

Retornar valor

Se um aplicativo processa essa mensagem, ele deve retornar zero.

Comentários

Use o código a seguir para obter a posição horizontal e vertical:

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

Conforme mencionado acima, a coordenada x está na ordem baixa **a menos** que o valor de retorno; a coordenada y está em ordem alta curta **(ambas** representam valores assinados porque podem receber valores negativos em sistemas com vários monitores). Se o valor de retorno for atribuído a uma variável, você poderá usar a macro MAKEPOINTS para obter uma estrutura POINTS do valor de retorno. Você também pode usar a macro GET _ X _ LPARAM ou GET Y _ _ LPARAM para extrair a coordenada x ou y.

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor porque essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas, e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades sem sinal.

Para detectar que a tecla ALT foi pressionada, verifique se GetKeyState com _ **MENU VK** < 0. Observe que isso não deve ser GetAsyncKeyState.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (inclua Windowsx.h)

Confira também

Referência

GET _ X _ LPARAM

GET _ Y _ LPARAM

GetCapture

Getkeystate

Setcapture

WM _ RBUTTONDBLCLK

WM _ RBUTTONUP

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

PONTOS

Mensagem _ WM RBUTTONUP

15/04/2022 • 2 minutes to read

Postado quando o usuário libera o botão direito do mouse enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.

Uma janela recebe essa mensagem por meio de **sua função WindowProc.**

```
#define WM_RBUTTONUP                    0x0205
```

Parâmetros

wParam

Indica se várias chaves virtuais estão inoadas. Esse parâmetro pode ser um ou mais dos valores a seguir.

VALOR	SIGNIFICADO
MK _ CONTROLE 0x0008	A tecla CTRL está inocizada.
MK _ LBUTTON 0x0001	O botão esquerdo do mouse está ino mouse.
MK _ MBUTTON 0x0010	O botão do meio do mouse está ino mouse.
MK _ RBUTTON 0x0002	A tecla SHIFT está inobada.
MK _ XBUTTON1 0x0020	O primeiro botão X está ino mouse.
MK _ XBUTTON2 0x0040	O segundo botão X está ino mouse.

lParam

A palavra de ordem baixa especifica a coordenada X do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

A palavra de ordem alta especifica a coordenada y do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

Retornar valor

Se um aplicativo processa essa mensagem, ele deve retornar zero.

Comentários

Use o código a seguir para obter a posição horizontal e vertical:

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

Conforme mencionado acima, a coordenada x está na ordem baixa **a menos** que o valor de retorno; a coordenada y está em ordem alta curta **(ambos** representam valores assinados porque podem receber valores negativos em sistemas com vários monitores). Se o valor de retorno for atribuído a uma variável, você poderá usar a macro MAKEPOINTS para obter uma estrutura POINTS do valor de retorno. Você também pode usar a macro GET _ X _ LPARAM ou GET Y _ _ LPARAM para extrair a coordenada x ou y.

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor porque essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas, e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades sem sinal.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (inclua Windowsx.h)

Confira também

Referência

GET _ X _ LPARAM

GET _ Y _ LPARAM

GetCapture

Setcapture

WM _ RBUTTONDBLCLK

WM _ RBUTTONDOWN

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

PONTOS

Mensagem WM _ XBUTTONDBLCLK

15/04/2022 • 3 minutes to read

Postado quando o usuário clica duas vezes no primeiro ou segundo botão X enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.

Uma janela recebe essa mensagem por meio de **sua função WindowProc.**

```
#define WM_XBUTTONDBLCLK                0x020D
```

Parâmetros

wParam

A palavra de ordem baixa indica se várias chaves virtuais estão inotivas. Pode ser um ou mais dos valores a seguir.

VALOR	SIGNIFICADO
MK _ CONTROLE 0x0008	A tecla CTRL está inocizada.
MK _ LBUTTON 0x0001	O botão esquerdo do mouse está ino mouse.
MK _ MBUTTON 0x0010	O botão do meio do mouse está ino mouse.
MK _ RBUTTON 0x0002	O botão direito do mouse está ino mouse.
MK _ SHIFT 0x0004	A tecla SHIFT está inobada.
MK _ XBUTTON1 0x0020	O primeiro botão X está ino mouse.
MK _ XBUTTON2 0x0040	O segundo botão X está ino mouse.

A palavra de ordem alta indica qual botão foi clicado duas vezes. Pode ser um dos seguintes valores.

VALOR	SIGNIFICADO
XBUTTON1 0x0001	O primeiro botão X foi clicado duas vezes.
XBUTTON2 0x0002	O segundo botão X foi clicado duas vezes.

lParam

A palavra de ordem baixa especifica a coordenada X do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

A palavra de ordem alta especifica a coordenada y do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

Retornar valor

Se um aplicativo processa essa mensagem, ele deve retornar **TRUE.** Para obter mais informações sobre como processar o valor de retorno, consulte a seção Comentários.

Comentários

Use o código a seguir para obter as informações no *parâmetro wParam:*

```
fwKeys = GET_KEYSTATE_WPARAM (wParam);
fwButton = GET_XBUTTON_WPARAM (wParam);
```

Use o código a seguir para obter a posição horizontal e vertical:

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

Conforme mencionado acima, a coordenada x está na ordem baixa **a menos** que o valor de retorno; a coordenada y está em ordem alta curta **(ambas** representam valores assinados porque podem receber valores negativos em sistemas com vários monitores). Se o valor de retorno for atribuído a uma variável, você poderá usar a macro MAKEPOINTS para obter uma estrutura POINTS do valor de retorno. Você também pode usar a macro GET _ X _ LPARAM ou GET Y _ _ LPARAM para extrair a coordenada x ou y.

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor porque essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas, e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades sem sinal.

Somente as janelas que têm o estilo _ **DBLCLKS CS** podem receber mensagens **WM _ XBUTTONDBLCLK,** que o sistema gera sempre que o usuário pressiona, libera e pressiona novamente um botão X dentro do limite de tempo de clique duplo do sistema. Clicar duas vezes em um desses botões gera quatro mensagens: WM _ XBUTTONDOWN, WM _ XBUTTONUP, **WM _ XBUTTONDBLCLK** e **WM _ XBUTTONUP** novamente.

Ao contrário das mensagens WM _ LBUTTONDBLCLK, WM _ MBUTTONDBLCLKe WM _

RBUTTONDBLCLK, um aplicativo deverá retornar **TRUE** dessa mensagem se processá-la. Isso permitirá que o software que simula essa mensagem em sistemas Windows anteriores ao Windows 2000 determine se o procedimento de janela processou a mensagem ou chamou **DefWindowProc** para processá-la.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (inclua Windowsx.h)

Confira também

Referência

Defwindowproc

GET _ KEYSTATE _ WPARAM

GET _ X _ LPARAM

GET _ XBUTTON _ WPARAM

GET _ Y _ LPARAM

GetCapture

GetDoubleClickTime

SetDoubleClickTime

WM _ XBUTTONDOWN

WM _ XBUTTONUP

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

PONTOS

Mensagem WM _ XBUTTONDOWN

15/04/2022 • 2 minutes to read

Postado quando o usuário pressiona o primeiro ou segundo botão X enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.

Uma janela recebe essa mensagem por meio de **sua função WindowProc.**

```
#define WM_XBUTTONDOWN                  0x020B
```

Parâmetros

wParam

A palavra de ordem baixa indica se várias chaves virtuais estão inotivas. Pode ser um ou mais dos valores a seguir.

VALOR	SIGNIFICADO
MK _ CONTROLE 0x0008	A tecla CTRL está inocizada.
MK _ LBUTTON 0x0001	O botão esquerdo do mouse está ino mouse.
MK _ MBUTTON 0x0010	O botão do meio do mouse está ino mouse.
MK _ RBUTTON 0x0002	O botão direito do mouse está ino mouse.
MK _ SHIFT 0x0004	A tecla SHIFT está inobada.
MK _ XBUTTON1 0x0020	O primeiro botão X está ino mouse.
MK _ XBUTTON2 0x0040	O segundo botão X está ino mouse.

A palavra de ordem alta indica qual botão foi clicado. Pode ser um dos seguintes valores.

VALOR	SIGNIFICADO
XBUTTON1 0x0001	O primeiro botão X foi clicado.
XBUTTON2 0x0002	O segundo botão X foi clicado.

lParam

A palavra de ordem baixa especifica a coordenada X do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

A palavra de ordem alta especifica a coordenada y do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

Retornar valor

Se um aplicativo processa essa mensagem, ele deve retornar **TRUE.** Para obter mais informações sobre como processar o valor de retorno, consulte a seção Comentários.

Comentários

Use o código a seguir para obter as informações no *parâmetro wParam:*

```
fwKeys = GET_KEYSTATE_WPARAM (wParam);
fwButton = GET_XBUTTON_WPARAM (wParam);
```

Use o código a seguir para obter a posição horizontal e vertical:

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

Conforme mencionado acima, a coordenada x está na ordem baixa **a menos** que o valor de retorno; a coordenada y está em ordem alta curta **(ambos** representam valores assinados porque podem receber valores negativos em sistemas com vários monitores). Se o valor de retorno for atribuído a uma variável, você poderá usar a macro MAKEPOINTS para obter uma estrutura POINTS do valor de retorno. Você também pode usar a macro GET _ X _ LPARAM ou GET Y _ _ LPARAM para extrair a coordenada x ou y.

> **IMPORTANT**
> Não use as macros **LOWORD** ou **HIWORD** para extrair as coordenadas x e y da posição do cursor porque essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas, e **LOWORD** e **HIWORD** tratam as coordenadas como quantidades sem sinal.

Ao contrário das mensagens WM _ LBUTTONDOWN, WM _ MBUTTONDOWNe WM _ RBUTTONDOWN, um aplicativo deverá retornar **TRUE** dessa mensagem se processá-lo. Isso permite que o software que simula essa mensagem em sistemas Windows anteriores ao Windows 2000 determine se o procedimento de janela processou a mensagem ou chamou DefWindowProc para processá-la.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	Winuser.h (inclua Windowsx.h)

Confira também

Referência

GET _ KEYSTATE _ WPARAM

GET _ X _ LPARAM

GET _ XBUTTON _ WPARAM

GET _ Y _ LPARAM

GetCapture

Setcapture

WM _ XBUTTONDBLCLK

WM _ XBUTTONUP

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

PONTOS

Mensagem do WM _ XBUTTONUP

15/04/2022 • 2 minutes to read

Postado quando o usuário libera o primeiro ou o segundo botão X enquanto o cursor está na área do cliente de uma janela. Se o mouse não for capturado, a mensagem será postada na janela abaixo do cursor. Caso contrário, a mensagem será postada na janela que capturou o mouse.

Uma janela recebe essa mensagem por meio de sua função **WindowProc** .

```
#define WM_XBUTTONUP                    0x020C
```

Parâmetros

wParam

A palavra de ordem inferior indica se várias chaves virtuais estão inativas. Pode ser um ou mais dos valores a seguir.

VALOR	SIGNIFICADO
MK _ 0X0008 de controle	A tecla CTRL está inoperante.
MK _ LBUTTON 0x0001	O botão esquerdo do mouse está inativo.
MK _ MBUTTON 0x0010	O botão do meio do mouse está inativo.
MK _ RBUTTON 0x0002	O botão direito do mouse está inativo.
MK _ SHIFT 0x0004	A tecla SHIFT está inoperante.
MK _ XBUTTON1 0x0020	O primeiro botão X está inoperante.
MK _ XBUTTON2 0x0040	O segundo botão X está inoperante.

A palavra de ordem superior indica qual botão foi liberado. Pode ser um dos seguintes valores:

VALOR	SIGNIFICADO
XButton1 0x0001	O primeiro botão X foi liberado.
XButton2 0x0002	O segundo botão X foi liberado.

lParam

A palavra de ordem inferior Especifica a coordenada x do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

A palavra de ordem superior especifica a coordenada y do cursor. A coordenada é relativa ao canto superior esquerdo da área do cliente.

Retornar valor

Se um aplicativo processar essa mensagem, ele deverá retornar **true**. Para obter mais informações sobre como processar o valor de retorno, consulte a seção comentários.

Comentários

Use o código a seguir para obter as informações no parâmetro *wParam* :

```
fwKeys = GET_KEYSTATE_WPARAM (wParam);
fwButton = GET_XBUTTON_WPARAM (wParam);
```

Use o código a seguir para obter a posição horizontal e vertical:

```
xPos = GET_X_LPARAM(lParam);
yPos = GET_Y_LPARAM(lParam);
```

Conforme observado acima, a coordenada x está **na ordem inferior** do valor de retorno; a coordenada y está no **intervalo** de ordem superior (ambos representam valores *assinados* porque podem receber valores negativos em sistemas com vários monitores). Se o valor de retorno for atribuído a uma variável, você poderá usar a macro MAKEPOINTS para obter uma estrutura de **pontos** do valor de retorno. Você também pode usar a macro Get _ X _ lParam ou Get _ y _ lParam para extrair a coordenada x ou Y.

> **IMPORTANT**
> Não use as macros LOWORD ou HIWORD para extrair as coordenadas x e y da posição do cursor, pois essas macros retornam resultados incorretos em sistemas com vários monitores. Sistemas com vários monitores podem ter coordenadas x e y negativas e LOWORD e HIWORD tratam as coordenadas como quantidades não assinadas.

Ao contrário das mensagens do WM _ LBUTTONUP, do WM _ MBUTTONUPe do WM _ RBUTTONUP , um aplicativo deve retornar **true** dessa mensagem se a processar. isso permitirá que o software que simula essa mensagem em sistemas Windows anteriores a Windows 2000 para determinar se o procedimento de janela processou a mensagem ou se chamou DefWindowProc para processá-la.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows 2000 Professional [somente aplicativos da área de trabalho]
Servidor mínimo com suporte	Windows 2000 Server [somente aplicativos da área de trabalho]
Cabeçalho	WinUser. h (incluir Windowsx. h)

Confira também

Referência

OBTER _ _ wParam KeyState

OBTER _ X _ lParam

OBTER _ _ wParam XBUTTON

OBTER _ _ lParam Y

GetCapture

SetCapture

XBUTTONDBLCLK do WM _

XBUTTONDOWN do WM _

Conceitual

Entrada do mouse

Outros recursos

MAKEPOINTS

FAIXAS

Estruturas de entrada do mouse

15/04/2022 • 2 minutes to read

Nesta seção

- MOUSEMOVEPOINT
- TRACKMOUSEEVENT

Entrada bruta

15/04/2022 • 2 minutes to read

Esta seção descreve como o sistema fornece entrada bruta para seu aplicativo e como um aplicativo recebe e processa essa entrada. Às vezes, a entrada bruta é chamada de entrada genérica.

Nesta seção

NOME	DESCRIÇÃO
Sobre a entrada bruta	Discute a entrada de usuários de dispositivos como joysticks, telas de toque e microfones.
Usando a entrada bruta	Fornece código de exemplo para tarefas relacionadas à entrada bruta.
Referência de entrada bruta	Contém a referência de API.

Funções

NOME	DESCRIÇÃO
DefRawInputProc	Chama o procedimento de entrada bruto padrão para fornecer o processamento padrão para todas as mensagens de entrada não processadas que um aplicativo não processa. Essa função garante que cada mensagem seja processada. **DefRawInputProc** é chamado com os mesmos parâmetros recebidos pelo procedimento de janela.
GetRawInputBuffer	Executa uma leitura em buffer dos dados de entrada brutos.
GetRawInputData	Obtém a entrada bruta do dispositivo especificado.
GetRawInputDeviceInfo	Obtém informações sobre o dispositivo de entrada bruto.
GetRawInputDeviceList	Enumera os dispositivos de entrada brutos anexados ao sistema.
GetRegisteredRawInputDevices	Obtém as informações sobre os dispositivos de entrada brutos para o aplicativo atual.
RegisterRawInputDevices	Registra os dispositivos que fornecem os dados de entrada brutos.

Macros

NOME	DESCRIÇÃO
OBTER _ _ wParam do código rawinput _	Obtém o código de entrada do *wParam* na **_ entrada do WM**.

NOME	DESCRIÇÃO
NEXTRAWINPUTBLOCK	Obtém o local da próxima estrutura em uma matriz de estruturas rawinput .

Notificações

NOME	DESCRIÇÃO
entrada do WM _	Enviado para a janela que está obtendo entrada bruta.
_alteração do _ dispositivo de entrada do WM _	Enviado para a janela que está registrada para receber entrada bruta.

Estruturas

NOME	DESCRIÇÃO
RAWHID	Descreve o formato da entrada bruta de um dispositivos de interface humana (HID).
RAWINPUT	Contém a entrada bruta de um dispositivo.
RAWINPUTDEVICE	Define informações para os dispositivos de entrada brutos.
RAWINPUTDEVICELIST	Contém informações sobre um dispositivo de entrada bruto.
RAWINPUTHEADER	Contém as informações de cabeçalho que fazem parte dos dados de entrada brutos.
RAWKEYBOARD	Contém informações sobre o estado do teclado.
RAWMOUSE	Contém informações sobre o estado do mouse.
_informações do dispositivo RID _	Define os dados de entrada brutos provenientes de qualquer dispositivo.
informações de dispositivo de RID _ _ _ HID	Define os dados brutos de entrada provenientes do HID especificado.
_teclado de _ informações do dispositivo RID _	Define os dados brutos de entrada provenientes do teclado especificado.
_mouse de _ informações _ do dispositivo RID	Define os dados brutos de entrada provenientes do mouse especificado.

Sobre a entrada bruta

15/04/2022 • 5 minutes to read

Há muitos dispositivos de entrada de usuário ao lado do teclado e do mouse tradicionais. Por exemplo, a entrada do usuário pode vir de um joystick, uma tela sensível ao toque, um microfone ou outros dispositivos que permitem grande flexibilidade na entrada do usuário. Esses dispositivos são coletivamente conhecidos como HIDs (dispositivos de interface humana). A API de entrada bruta fornece uma maneira estável e robusta para que os aplicativos aceitem a entrada bruta de qualquer HID, incluindo o teclado e o mouse.

Esta seção contém os seguintes tópicos:

- Modelo de entrada bruto
- Registro de entrada bruta
- Lendo entrada bruta

Modelo de entrada bruto

Anteriormente, o teclado e o mouse normalmente geraram dados de entrada. O sistema interpretou os dados provenientes desses dispositivos de forma a eliminar os detalhes específicos do dispositivo das informações brutas. Por exemplo, o teclado gera o código de verificação específico do dispositivo, mas o sistema fornece um aplicativo com o código de chave virtual. Além de ocultar os detalhes da entrada bruta, o Gerenciador de janelas não dava suporte a todos os novos HIDs. Para obter a entrada do HIDs sem suporte, um aplicativo tinha que fazer muitas coisas: abrir o dispositivo, gerenciar o modo compartilhado, ler periodicamente o dispositivo ou configurar a porta de conclusão de e/s e assim por diante. O modelo de entrada bruto e as APIs associadas foram desenvolvidas para permitir o acesso simples à entrada bruta de todos os dispositivos de entrada, incluindo o teclado e o mouse.

o modelo de entrada bruto é diferente do modelo de entrada de Windows original para o teclado e o mouse. No modelo de entrada original, um aplicativo recebe entrada independente de dispositivo na forma de mensagens que são enviadas ou postadas em suas janelas, como **WM _ Char**, **WM _ MOUSEMOVE**e **WM _ APPCOMMAND**. Por outro lado, para a entrada bruta, um aplicativo deve registrar os dispositivos do qual deseja obter dados. Além disso, o aplicativo obtém a entrada bruta por meio da mensagem de **_ entrada do WM** .

Há várias vantagens para o modelo de entrada bruto:

- Um aplicativo não precisa detectar ou abrir o dispositivo de entrada.
- Um aplicativo obtém os dados diretamente do dispositivo e processa os dados para suas necessidades.
- Um aplicativo pode distinguir a origem da entrada mesmo que ela seja do mesmo tipo de dispositivo. Por exemplo, dois dispositivos de mouse.
- Um aplicativo gerencia o tráfego de dados especificando dados de uma coleção de dispositivos ou apenas tipos de dispositivos específicos.
- Os dispositivos HID podem ser usados à medida que se tornam disponíveis no Marketplace, sem aguardar novos tipos de mensagem ou um sistema operacional atualizado para ter novos comandos no **WM _ APPCOMMAND**.

Observe que o **WM _ APPCOMMAND** fornece alguns dispositivos HID. No entanto, o **WM _ APPCOMMAND** é um evento de entrada independente de dispositivo de nível superior, enquanto a **_ entrada do WM** envia dados brutos de baixo nível que são específicos para um dispositivo.

Registro de entrada bruta

Por padrão, nenhum aplicativo recebe entrada bruta. Para receber a entrada bruta de um dispositivo, um aplicativo deve registrar o dispositivo.

Para registrar dispositivos, um aplicativo primeiro cria uma matriz de estruturas **RAWINPUTDEVICE** que especificam a coleção de nível superior (TLC) para os dispositivos que ele deseja. O TLC é definido por uma página de uso (a classe do dispositivo) e uma ID de uso (o dispositivo dentro da classe). Por exemplo, para obter o teclado TLC, defina UsagePage = 0x01 e Usageid = 0x06. O aplicativo chama **RegisterRawInputDevices** para registrar os dispositivos.

Observe que um aplicativo pode registrar um dispositivo que não está anexado no momento ao sistema. quando esse dispositivo estiver anexado, o gerenciador de Windows enviará automaticamente a entrada bruta para o aplicativo. Para obter a lista de dispositivos de entrada brutos no sistema, um aplicativo chama **GetRawInputDeviceList**. Usando o *hDevice* dessa chamada, um aplicativo chama **GetRawInputDeviceInfo** para obter as informações do dispositivo.

Por meio do membro **dwFlags** de **RAWINPUTDEVICE**, um aplicativo pode selecionar os dispositivos a serem escutados e também aqueles que deseja ignorar. Por exemplo, um aplicativo pode solicitar entrada de todos os dispositivos de telefonia, exceto para as máquinas de resposta. Para obter o código de exemplo, consulte registrando para entrada bruta.

Observe que o mouse e o teclado também são HIDs, portanto, os dados deles podem vir por meio da **_ entrada do WM** de mensagem HID e de mensagens tradicionais. Um aplicativo pode selecionar um dos métodos pela seleção apropriada de sinalizadores em **RAWINPUTDEVICE**.

Para obter o status do registro de um aplicativo, chame **GetRegisteredRawInputDevices** a qualquer momento.

Lendo entrada bruta

Um aplicativo recebe entrada bruta de qualquer HID cuja coleção de nível superior (TLC) corresponde a um TLC do registro. Quando um aplicativo recebe entrada bruta, sua fila de mensagens Obtém uma mensagem de **_ entrada do WM** e o sinalizador de status da fila **QS _ rawinput** é definido (**QS _ entrada** também inclui esse sinalizador). Um aplicativo pode receber dados quando estiver em primeiro plano e quando estiver em segundo plano.

Há duas maneiras de ler os dados brutos: o método sem buffer (ou padrão) e o método em buffer. O método sem buffer obtém os dados brutos de uma estrutura **rawinput** por vez e é adequado para muitos HIDS. Aqui, o aplicativo chama **GetMessage** para obter a mensagem de **_ entrada do WM** . Em seguida, o aplicativo chama **GetRawInputData** usando o identificador **rawinput** contido na **_ entrada do WM**. Para obter um exemplo, consulte fazendo uma leitura padrão de entrada bruta.

Por outro lado, o método em buffer obtém uma matriz de estruturas **rawinput** de cada vez. Isso é fornecido para dispositivos que podem produzir grandes quantidades de entrada bruta. Nesse método, o aplicativo chama **GetRawInputBuffer** para obter uma matriz de estruturas **rawinput** . Observe que a macro **NEXTRAWINPUTBLOCK** é usada para atravessar uma matriz de estruturas **rawinput** . Para obter um exemplo, consulte fazendo uma leitura em buffer de entrada bruta.

Para interpretar a entrada bruta, são necessárias informações detalhadas sobre o HIDs. Um aplicativo obtém as informações do dispositivo chamando **GetRawInputDeviceInfo** com o identificador do dispositivo. Esse identificador pode ser proveniente da **_ entrada do WM** ou do membro **hDevice** de **RAWINPUTHEADER**.

Usando a entrada bruta

15/04/2022 • 2 minutes to read

Esta seção inclui código de exemplo para as seguintes finalidades:

- Registrando para entrada bruta
  - Exemplo 1
  - Exemplo 2
- Executando uma leitura padrão de entrada bruta
- Executando uma leitura em buffer de entrada bruta

Registrando para entrada bruta

Exemplo 1

Neste exemplo, um aplicativo especifica a entrada bruta dos controladores de jogo (pads de jogo e de gamepads) e todos os dispositivos fora da página de uso de telefonia, exceto os computador de resposta.

```
RAWINPUTDEVICE Rid[4];

Rid[0].usUsagePage = 0x01;          // HID_USAGE_PAGE_GENERIC
Rid[0].usUsage = 0x05;              // HID_USAGE_GENERIC_GAMEPAD
Rid[0].dwFlags = 0;                 // adds game pad
Rid[0].hwndTarget = 0;

Rid[1].usUsagePage = 0x01;          // HID_USAGE_PAGE_GENERIC
Rid[1].usUsage = 0x04;              // HID_USAGE_GENERIC_JOYSTICK
Rid[1].dwFlags = 0;                 // adds joystick
Rid[1].hwndTarget = 0;

Rid[2].usUsagePage = 0x0B;          // HID_USAGE_PAGE_TELEPHONY
Rid[2].usUsage = 0x00;
Rid[2].dwFlags = RIDEV_PAGEONLY;    // adds all devices from telephony page
Rid[2].hwndTarget = 0;

Rid[3].usUsagePage = 0x0B;          // HID_USAGE_PAGE_TELEPHONY
Rid[3].usUsage = 0x02;              // HID_USAGE_TELEPHONY_ANSWERING_MACHINE
Rid[3].dwFlags = RIDEV_EXCLUDE;     // excludes answering machines
Rid[3].hwndTarget = 0;

if (RegisterRawInputDevices(Rid, 4, sizeof(Rid[0])) == FALSE)
{
    //registration failed. Call GetLastError for the cause of the error.
}
```

Exemplo 2

Neste exemplo, um aplicativo deseja entrada bruta do teclado e do mouse, mas deseja ignorar mensagens herdadas de janela do mouse e teclado (que seriam fornecidas do mesmo teclado e mouse).

```
RAWINPUTDEVICE Rid[2];

Rid[0].usUsagePage = 0x01;          // HID_USAGE_PAGE_GENERIC
Rid[0].usUsage = 0x02;              // HID_USAGE_GENERIC_MOUSE
Rid[0].dwFlags = RIDEV_NOLEGACY;    // adds mouse and also ignores legacy mouse messages
Rid[0].hwndTarget = 0;

Rid[1].usUsagePage = 0x01;          // HID_USAGE_PAGE_GENERIC
Rid[1].usUsage = 0x06;              // HID_USAGE_GENERIC_KEYBOARD
Rid[1].dwFlags = RIDEV_NOLEGACY;    // adds keyboard and also ignores legacy keyboard messages
Rid[1].hwndTarget = 0;

if (RegisterRawInputDevices(Rid, 2, sizeof(Rid[0])) == FALSE)
{
    //registration failed. Call GetLastError for the cause of the error
}
```

Executando uma leitura padrão de entrada bruta

Este exemplo mostra como um aplicativo faz uma leitura não padronada (ou padrão) de entrada bruta de um teclado ou hid (mouse dispositivos de interface humana) e, em seguida, imprime várias informações do dispositivo.

```
case WM_INPUT:
{
    UINT dwSize;

    GetRawInputData((HRAWINPUT)lParam, RID_INPUT, NULL, &dwSize, sizeof(RAWINPUTHEADER));
    LPBYTE lpb = new BYTE[dwSize];
    if (lpb == NULL)
    {
        return 0;
    }

    if (GetRawInputData((HRAWINPUT)lParam, RID_INPUT, lpb, &dwSize, sizeof(RAWINPUTHEADER)) != dwSize)
         OutputDebugString (TEXT("GetRawInputData does not return correct size !\n"));

    RAWINPUT* raw = (RAWINPUT*)lpb;

    if (raw->header.dwType == RIM_TYPEKEYBOARD)
    {
        hResult = StringCchPrintf(szTempOutput, STRSAFE_MAX_CCH,
            TEXT(" Kbd: make=%04x Flags:%04x Reserved:%04x ExtraInformation:%08x, msg=%04x VK=%04x \n"),
            raw->data.keyboard.MakeCode,
            raw->data.keyboard.Flags,
            raw->data.keyboard.Reserved,
            raw->data.keyboard.ExtraInformation,
            raw->data.keyboard.Message,
            raw->data.keyboard.VKey);
        if (FAILED(hResult))
        {
        // TODO: write error handler
        }
        OutputDebugString(szTempOutput);
    }
    else if (raw->header.dwType == RIM_TYPEMOUSE)
    {
        hResult = StringCchPrintf(szTempOutput, STRSAFE_MAX_CCH,
            TEXT("Mouse: usFlags=%04x ulButtons=%04x usButtonFlags=%04x usButtonData=%04x ulRawButtons=%04x
lLastX=%04x lLastY=%04x ulExtraInformation=%04x\r\n"),
            raw->data.mouse.usFlags,
            raw->data.mouse.ulButtons,
            raw->data.mouse.usButtonFlags,
            raw->data.mouse.usButtonData,
            raw->data.mouse.ulRawButtons,
            raw->data.mouse.lLastX,
            raw->data.mouse.lLastY,
            raw->data.mouse.ulExtraInformation);

        if (FAILED(hResult))
        {
        // TODO: write error handler
        }
        OutputDebugString(szTempOutput);
    }

    delete[] lpb;
    return 0;
}
```

Executando uma leitura em buffer de entrada bruta

Este exemplo mostra como um aplicativo faz uma leitura em buffer de entrada bruta de um HID genérico.

```
case MSG_GETRIBUFFER: // Private message
    {
    UINT cbSize;
    Sleep(1000);

    VERIFY(GetRawInputBuffer(NULL, &cbSize, sizeof(RAWINPUTHEADER)) == 0);
    cbSize *= 16; // up to 16 messages
    Log(_T("Allocating %d bytes"), cbSize);
    PRAWINPUT pRawInput = (PRAWINPUT)malloc(cbSize);
    if (pRawInput == NULL)
    {
        Log(_T("Not enough memory"));
        return;
    }

    for (;;)
    {
        UINT cbSizeT = cbSize;
        UINT nInput = GetRawInputBuffer(pRawInput, &cbSizeT, sizeof(RAWINPUTHEADER));
        Log(_T("nInput = %d"), nInput);
        if (nInput == 0)
        {
            break;
        }

        ASSERT(nInput > 0);
        PRAWINPUT* paRawInput = (PRAWINPUT*)malloc(sizeof(PRAWINPUT) * nInput);
        if (paRawInput == NULL)
        {
            Log(_T("paRawInput NULL"));
            break;
        }

        PRAWINPUT pri = pRawInput;
        for (UINT i = 0; i < nInput; ++i)
        {
            Log(_T(" input[%d] = @%p"), i, pri);
            paRawInput[i] = pri;
            pri = NEXTRAWINPUTBLOCK(pri);
        }

        free(paRawInput);
    }
    free(pRawInput);
}
```

Referência de entrada bruta

15/04/2022 • 2 minutes to read

Nesta seção

- Funções de entrada brutas
- Macros de entrada brutas
- Notificações de entrada brutas
- Estruturas de entrada brutas

Funções de entrada brutas

15/04/2022 • 2 minutes to read

Nesta seção

- DefRawInputProc
- GetRawInputBuffer
- GetRawInputData
- GetRawInputDeviceInfo
- GetRawInputDeviceList
- GetRegisteredRawInputDevices
- RegisterRawInputDevices

Macros de Entrada Bruta

15/04/2022 • 2 minutes to read

Nesta seção

- GET _ RAWINPUT _ CODE _ WPARAM
- NEXTRAWINPUTBLOCK

Notificações de entrada brutas

15/04/2022 • 2 minutes to read

Nesta seção

- entrada do WM _
- _alteração do _ dispositivo de entrada do WM _

Mensagem WM _ INPUT

15/04/2022 • 2 minutes to read

Enviado para a janela que está recebendo entrada bruta.

Uma janela recebe essa mensagem por meio de **sua função WindowProc.**

```
#define WM_INPUT 0x00FF
```

Parâmetros

wParam

O código de entrada. Use **a macro _ _ _ WPARAM GET RAWINPUT CODE** para obter o valor.

Pode ser um dos seguintes valores:

VALOR	SIGNIFICADO
RIM _ ENTRADA 0	A entrada ocorreu enquanto o aplicativo estava em primeiro plano. O aplicativo deve chamar **DefWindowProc** para que o sistema possa executar a limpeza.
RIM _ INPUTSINK 1	A entrada ocorreu enquanto o aplicativo não estava em primeiro plano.

lParam

Um **alça HRAWINPUT** para a **estrutura RAWINPUT** que contém a entrada bruta do dispositivo. Para obter os dados brutos, use esse handle na chamada para **GetRawInputData.**

Retornar valor

Se um aplicativo processa essa mensagem, ele deve retornar zero.

Comentários

A entrada bruta está disponível somente quando o aplicativo **chama RegisterRawInputDevices com** especificações de dispositivo válidas.

Requisitos

REQUISITO	VALOR
Cliente mínimo com suporte	Windows Somente [aplicativos da área de trabalho XP]
Servidor mínimo com suporte	Windows Somente aplicativos da área de trabalho server 2003 []

REQUISITO	VALOR
Cabeçalho	**Winuser.h (incluir Windows.h)**

Confira também

Referência

GetRawInputData

RegisterRawInputDevices

RAWINPUT

GET _ RAWINPUT _ CODE _ WPARAM

Conceitual

Entrada bruta

WM_INPUT_DEVICE_CHANGE mensagem

15/04/2022 • 2 minutes to read

Descrição

Enviado para a janela que está registrada para receber entrada bruta.

As notificações de entrada brutas estão disponíveis somente depois que o aplicativo chama RegisterRawInputDevices com RIDEV_DEVNOTIFY sinalizador.

Uma janela recebe essa mensagem por meio de sua função **WindowProc** .

```
#define WM_INPUT_DEVICE_CHANGE          0x00FE
```

Parâmetros

wParam

Tipo: **wParam**

Esse parâmetro pode usar um dos valores a seguir.

VALOR	SIGNIFICADO
chegada de GIDC _ 1	Um novo dispositivo foi adicionado ao sistema. Você pode chamar GetRawInputDeviceInfo para obter mais informações sobre o dispositivo.
remoção de GIDC _ 2	Um dispositivo foi removido do sistema.

lParam

Tipo: **lParam**

O **identificador** para o dispositivo de entrada bruto.

Retornar valor

Se um aplicativo processar essa mensagem, ele deverá retornar zero.

Confira também

Conceitual

Entrada bruta

Referência

RegisterRawInputDevices

Estrutura RAWINPUTDEVICE

Estruturas de entrada brutas

15/04/2022 • 2 minutes to read

Nesta seção

- RAWHID
- RAWINPUT
- RAWINPUTDEVICE
- RAWINPUTDEVICELIST
- RAWINPUTHEADER
- RAWKEYBOARD
- RAWMOUSE
- _informações do dispositivo RID _
- informações de dispositivo de RID _ _ _ HID
- _teclado de _ informações do dispositivo RID _
- _mouse de _ informações _ do dispositivo RID