WIN32 API (APPLICATION PROGRAMMING INTERFACE).

PROGRAMACIÓN 2

(LMAD)

MGTI ALEJANDRO VILLARREAL MOJICA

OBJECT: BUTTON

- Los Buttons son utilizados frecuentemente para ejecutar una acción en un programa, de forma similar a los botones Yes, No, y OK vistos en los MessageBox.
- Podemos crear nuestros propios botones para que hagan cualquier cosa que deseemos.
- Por ejemplo, podemos utilizar un botón para que abra una ventana nueva por separado.

CREANDO UN BOTÓN

- Para utilizar los botones o cualquier otro control, primero tenemos que definir un número único para cada objeto.
- Se utiliza la instrucción 'define' para establecer un ID.
- Generalmente la mejor manera de mantener las cláusulas 'define' es en la parte superior del código, para que su valor se conserve en todo el programa.
- Si se utiliza la creación de diálogos y objetos por diseño, el ID se generará de forma automática y se establecerá en el archivo 'resource.h'

```
#define IDC_MAIN_BUTTON 101
```

CREAR OBJETOS.

```
#define IDC_MAIN_BUTTON 101
#define IDC_MAIN_EDIT 102


HWND hButt, HEdit
```

MGTI Alejandro Villarreal --> alvillarreal@gmail.com

CREANDO UN BOTÓN

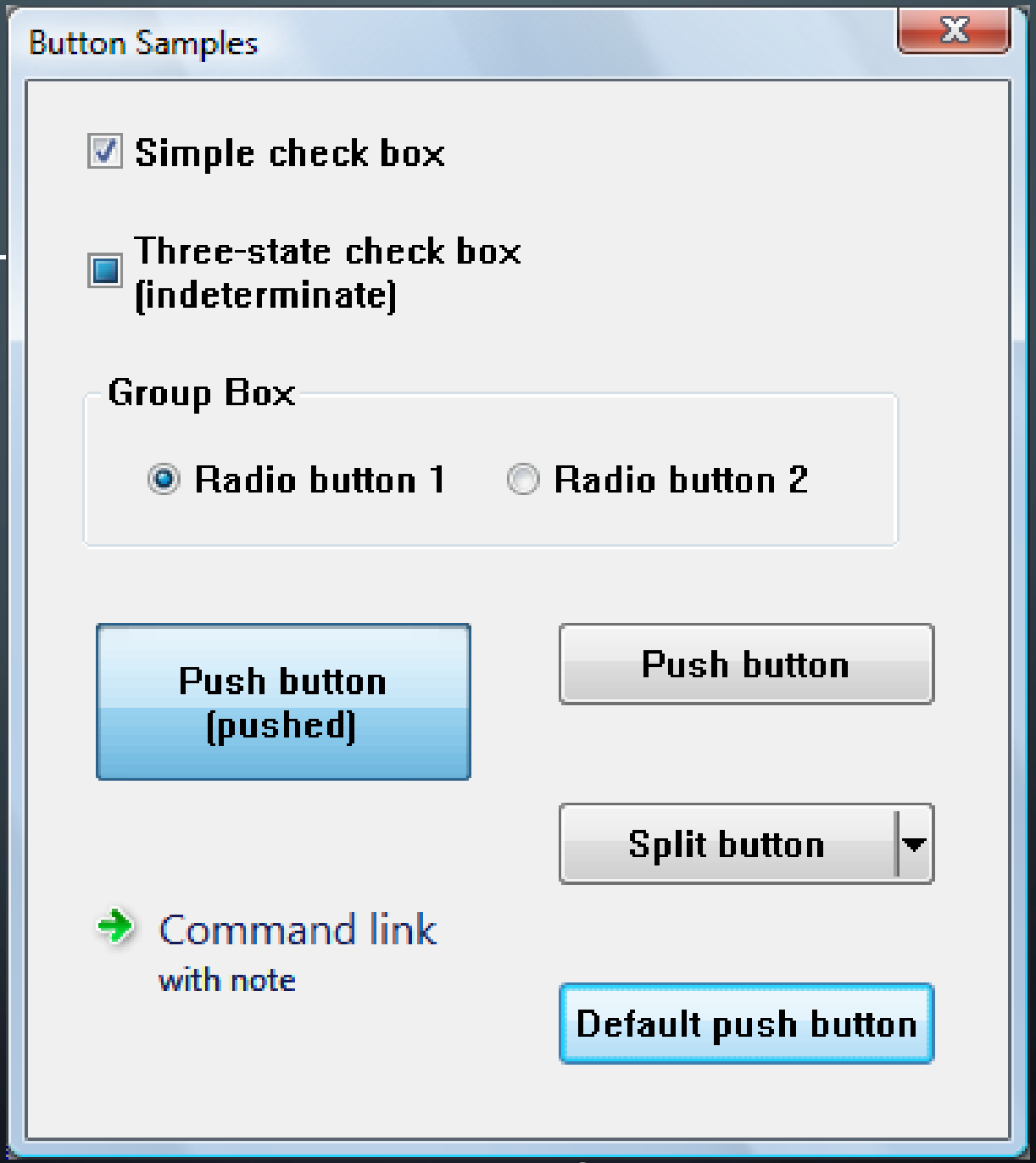
Al ser generados por diseño, no se cuenta con la variable tipo Windows Handle.

Por lo tanto se obtiene al momento de ejecutar el programa, con la función GetDlgItem

Ejemplo:

```
hButt = GetDlgItem(hDlg, IDC_MAIN_BUTTON);
```

WM_


Button Samples
Simple check box
Three-state check box (indeterminate)
Group Box
Radio button 1
Radio button 2
Push button (pushed)
Push button
Split button
Command link
with note
Default push button

CREANDO OTROS OBJETOS

El uso y manejo de otros objetos es muy similar, se requiere:

- ✓ ID
  - ✓ Identificador numérico positivo y único
- ✓ HWND
  - ✓ Variable tipo Window Handle, definida en la ejecución del programa

MENSAJE PARA ESTABLECER TEXTO EN UNA CAJITA DE TEXTO…

```
SendMessage(
        hEdit,
        WM_SETTEXT,
        NULL,
        (LPARAM)"Insert text here...");
```

MGTI Alejandro Villarreal --> alvillarreal@gmail.com

CALLBACK PROCEDURE: LEER UNA CAJA DE TEXTO Y PASARLO A UNA VARIABLE

```
case WM_COMMAND:
        switch(LOWORD(wParam))
            {
               case IDC_MAIN_BUTTON:
               {
                      char buffer[256];
                      SendMessage(hEdit, WM_GETTEXT,
                             sizeof(buffer)/sizeof(buffer[0]),
                             reinterpret_cast<LPARAM>(buffer));

                      MessageBox(NULL, buffer, "Information",
                                                    MB_ICONINFORMATION);
```

PONIENDO TODO EN USO

- Para hacer uso de nuestros objetos recién creados, tenemos que atender a los mensajes en nuestro procedimiento de CALLBACK.
- El mensaje que debemos aprovechar es: WM_COMMAND.
- Cada vez que se hace clic en un botón de tipo texto, el mensaje WM_COMMAND se está llamando.
- Es en el LOWORD del wParam quien nos dice exactamente lo que está sucediendo.
  - 16 bits

WM_COMMAND

```
case WM_COMMAND:
      switch(LOWORD(wParam))
      {
            case IDC_MAIN_BUTTON:
                  {
                  }
                  break;
      }
      break;
```

ANYTIME MESSAGES

- Los mensajes que hemos utilizado hasta ahora pertenecen a eventos específicos generados en un momento determinado por una ventana. A veces en medio de algo, es posible que desee enviar un mensaje, independientemente de lo que está pasando.
- Esto es possible con una función llamada: **SendMessage()**.
- La sintaxis de la función **SendMessage()** es la siguiente:

```
LRESULT SendMessage(HWND hWnd, UINT Msg,
WPARAM wParam, LPARAM lParam);
```

ANYTIME MESSAGES

```
LRESULT SendMessage(HWND hWnd, UINT Msg,
WPARAM wParam, LPARAM lParam);
```

- El parámetro *hWnd* representa al objeto o control que está enviando el mensaje.
- El parámetro *Msg* es el mensaje a ser enviado.
- Los valores de los parámetros *wParam* y *lParam* dependen del mensaje que está siendo enviado.

SENDMESSAGE() FUNCTION

- La ventaja de utilizar la función **SendMessage()** es que, al enviar este mensaje, apuntaría al procedimiento que puede realizar la tarea y esta función volvería sólo después de que su mensaje haya sido procesado.

SENDMESSAGE() FUNCTION

- Debido a que esta función (miembro) a veces se puede utilizar universalmente, es decir, por cualquier control u objeto, la aplicación no puede predecir el tipo de mensaje que **SendMessage()** está llevando a cabo.

SENDMESSAGE() FUNCTION

- Por lo tanto, (la desventaja probable es que) usted debe saber el (nombre o identidad del) mensaje que está enviando y debe proporcionar artículos de acompañamiento precisos

SENDMESSAGE() FUNCTION

- Para enviar un mensaje usando la función SendMessage(), debe saber qué mensaje está enviando y qué necesita para completar este mensaje.
- Por ejemplo, para cambiar el título de una ventana en cualquier momento, puede utilizar el mensaje WM_SETTEXT. La sintaxis a utilizar sería::

```
SendMessage(hWnd, WM_SETTEXT,
                  wParam, lParam);
```

SENDMESSAGE() FUNCTION

- Obviamente, tendrías que proporcionar el texto para el título que estás intentando cambiar. Esta cadena es llevada por el argumento lParam como una cadena. Para este mensaje, el wParam se ignora:

```
LPCTSTR strMsg = L"This message was sent";
SendMessage(hWnd, WM_SETTEXT, 0, (LPARAM)(LPCTSTR)strMsg);
```