WindowsAPI 関数について

Windows 標準で備わっている関数郡のことを API（Application Program Interface）といいます。この関数郡にはさまざまな種類がありこの関数を使えるようになるとプログラムの幅が広がります。

ここではその一部の関数の使い方を説明していきます。

API を使うには windows.h をインクルードしてください。関数によっては別のヘッダをインクルードする必要もあります。

*サウンド関連

使う場合は mmsystem.h をインクルードしてください（VC の場合 winmm.lib もリンクしてください）。

・ BOOL Beep(DWORD dwFreq, // 音の周波数
DWORD dwDuration // 音の持続時間);

これはビープ音を鳴らす関数です。第一引数に音の周波数を（37～32,767）設定します。第二引数には音の継続時間をミリ秒で設定します。

例:

Beep(400,200); //周波数400で0.2秒鳴らす。

・ BOOL PlaySound(LPCSTR pszSound, // 再生対象のサウンド
HMODULE hmod, // インスタンスハンドル
DWORD fdwSound // 再生フラグ);

これは単純に Wav を再生する関数です。pszSound に再生したい Wav ファイルを指定します。 hmod には、リソースから再生させる時にインスタンスハンドルを入れます。通常は NULL を指定します。fdwSound に各種の再生フラグを入れます。関数が成功すると、TRUE が返り、関数が失敗すると、FALSE が返ります。

再生フラグ

SND_ASYNC	サウンドを非同期再生し、サウンドが開始されると、PlaySound 関数は即座に制御を返します。
SND_FILENAME	pszSound パラメータは、ファイル名を表します。
SND_LOOP	サウンドを繰り返し再生します。pszSound パラメータで NULL を指定して PlaySound 関数を呼び出すと、サウンドが停止します。サウンドイベントを非同期再生するよう指示するために、SND_ASYNC と同時に指定しなければなりません。
SND_MEMORY	サウンドイベントのファイルは、メモリ内に既にロードされています。pszSound パラメータは、メモリ内のサウンドイメージへのポインタを表します。

SND_RESOURCE	pszSound パラメータは、リソース識別子を表します。
SND_SYNC	サウンドイベントを同期再生します。PlaySound 関数は、サウンドの再生が完了した後で制御を返します。

再生フラグは通常は SND_ASYNC SND_FILENAME SND_LOOP ぐらいしか使いません。

例:

```
//ループ再生
PlaySound(“BGM￥￥music.wav”,NULL,SND_FILENAME | SND_ASYNC |
                                                    SND_LOOP);
//停止
PlaySound(NULL,NULL,0);
```

・ MCIERROR mciSendString(

LPCTSTR lpszCommand, // コマンド文字列
LPTSTR lpszReturnString, // 情報を受け取るバッファ
UINT cchReturn, // バッファのサイズ
HANDLE hwndCallback // コールバックウィンドウのハンドル);

この関数はマルチメディアプレイヤーです。さまざまなファイルを再生することができます。lpszCommand に命令をコマンド文字列として代入します。その他の引数についてはあまり使うことはありません。戻り値には MCIERROR が返ります。判断には mciError 関数を使います。ここでは単純に再生するだけの例を載せて置きます。Mci は細かく設定すれば多くのことができるので興味がある人は聞きに来てください。

例:

```
//音楽の読み込み
mciSendString(“open BGM￥￥music.mp3 type MPEGVideo alias
                                    music”,NULL,0,NULL);
//再生
mciSendString(“play music”,NULL,0,NULL);
//停止
mciSendString(“stop music”,NULL,0,NULL);
//開放（音楽を読み込んだら必ず開放してください）
mciSendString(“close music”,NULL,0,NULL);
```

*マウス・キーボード関連

・ BOOL SetCursorPos(int X, // 水平位置 , int Y // 垂直位置);

マウスの座標を設定します。関数が成功すると0以外が返り、失敗すると0が返ります。

例:

```
SetCursorPos( 50,150 );
```

・ BOOL GetCursorPos(LPPOINT lpPoint // カーソルの位置);
マウスの位置を取得します。引数は POINT 型変数です。関数が成功すると0以外が返り、失敗すると0が返ります。

例:

```
POINT pt;
int x,y;
GetCursorPos( & pt );
x = pt.x; y = pt.y;
```

・ HWND SetFocus(HWND hWnd // ウィンドウのハンドル);
指定したウィンドウにフォーカスを設定します。成功するとアクティブになったウィンドウのハンドルが返ります。

例:

```
SetFocus(Form1->Handle);
```

・ void keybd_event(BYTE bVk, // 仮想キーコード
BYTE bScan, // ハードウェアスキャンコード
DWORD dwFlags, // 動作指定フラグ
ULONG_PTR dwExtraInfo // 追加情報);
この関数はフォーカスの当たっているウィンドウに仮想キーコードを送信します。仮想キーコードについてはネットなどで調べてください。

例:

```
// aキーの押し下げをシミュレートする。
keybd_event( 'A', 0, 0, 0 );
// aキーの離すをシミュレートする。
keybd_event('A', 0, KEYEVENTF_KEYUP, 0);
```

*ウィンドウ関連

・ HWND WindowFromPoint(POINT Point // 座標);
この関数はマウスのカーソルの下にあるウィンドウのハンドルを取得する関数です。第一引数にマウスポインタを格納した POINT 型変数を入れ、戻り値にウィンドウのハンドルが返ってきます。

例:

```
POINT pt;
HWND hwnd;

GetCursorPos(&pt);
Hwnd = WindowFromPoint(pt);
```

これで現在マウスカーソルの下にあるウィンドウのハンドルを取得できます。

・ int GetWindowText(
HWND hWnd, // ウィンドウまたはコントロールのハンドル
LPTSTR lpString, // テキストバッファ
int nMaxCount // コピーする最大文字数);
この関数は第一引数で指定したウィンドウの名前を取得します。lpString には char 型で nMaxCount には char 型の大きさを指定します。
例:
char str[1024];

GetWindowText(Form1->Handle, str, 1024);

Label1->Caption = str;

これで Label に Form のタイトルが入ります。

・ BOOL SetWindowText(
HWND hWnd, // ウィンドウまたはコントロールのハンドル
LPCTSTR lpString // タイトルまたはテキスト);
この関数は第一引数で指定したウィンドウのタイトルを第二引数で指定した文字に変えます。
例:
char str = “kswl にようこそ”;

SetWindowText(Form1->Handle, str);

*その他
・void Sleep(DWORD dwMilliseconds // スリープさせる時間);
この関数を呼び出すと dwMilliseconds で指定した時間（ミリ秒）だけプログラムをとめるとことができます。ただこの関数で長い時間を指定するとプログラムが固まって動かなくなるので注意してください。
例:
//0.5秒プログラムを止める
Sleep（500）;

・int MessageBox(HWND hWnd, // オーナーウィンドウのハンドル
PCTSTR pszText, // 表示文字列
PCTSTR pszCaption, // キャプションバーの表示文字列
UINT uType // メッセージボックスのタイプ);
この関数はメッセージボックスを呼び出します。hWnd には親ウィンドウのハンド

ルをわからなければ NULL を指定します。pszText にはメッセージボックス内に表示さしたい文字列を、pszCaption にはタイトルに表示させたい文字列を指定します。uType にはメッセージボックス内に設置するボタンの定数を指定します。

uType の定数一覧

※ボタン フラグ

定数	説明
MB_ABORTRETRYIGNORE	「中止」「再試行」「無視」ボタンを表示します。
MB_OK	「OK」ボタンを表示します。
MB_OKCANCEL	「OK」「キャンセル」ボタンを表示します。
MB_RETRYCANCEL	「再試行」「キャンセル」ボタンを表示します。
MB_YESNO	「はい」「いいえ」ボタンを表示します。
MB_YESNOCANCEL	「はい」「いいえ」「キャンセル」ボタンを表示します。

また uType には以下のフラグも一緒に使えます。

※アイコン フラグ

定数	説明
MB_ICONEXCLAMATION、MB_ICONWARNING	感嘆符「!」アイコンを表示します。
MB_ICONINFORMATION、MB_ICONASTERISK	吹き出しの中に小文字の「i」があるアイコンを表示します。
MB_ICONQUESTION	疑問符「?」アイコンを表示します。
MB_ICONSTOP、MB_ICONERROR、MB_ICONHAND	停止アイコンを表示します。

他にもフラグはありますがあまり使われないので省略します。

そして、関数が成功すると、押されたボタンを示す以下の定数のいずれかが返ります。失敗すると、0 が返ります。

定数	説明
IDABORT	「中止」ボタンが押されました。
IDCANCEL	「キャンセル」ボタンが押されました。

IDIGNORE	「無視」ボタンが押されました。
IDNO	「いいえ」ボタンが押されました。
IDOK	「OK ボタン」が押されました。
IDRETRY	「再試行」ボタンが押されました。
IDYES	「はい」ボタンが押されました。

例:

```
//メッセージボックスで OK かキャンセルのどちらが押されたかを
//判断するプログラム

int flag = MessageBox(NULL, “OK か Cancel を押してください”,
                “例”,MB_OKCANCEL | MB_ICONQUESTION);

if(flag == IDOK ){
        MessageBox(NULL, “OK が押されました”, “結果”, MB_OK);
}
else if(flag == IDCANCEL){
        MesssgeBox(NULL, “CNANCEL が押されました”, “結果”,
                                                MB_OK);
}
```