Process Box (PROCBOX.DLL)

(Version 2.00, 6/17/96)

Sometimes a lengthy process must be carried out by an application. Because Windows 3.1 is a non-preemptive multi-tasking environment, carrying out the process without occasionally returning control to Windows will hang the system. One approach is to build your own PeekMessage loop and make a small dialog box which contains a "CANCEL" button. Unfortunately, everytime you want to do this, you have to implement the code yourself. It would be nice if there was a Windows API that could do this for you.

PROCBOX.DLL encodes a set of functions which do just this. There are several ways to use the Process Box functions. The simplest mode involves a very simple single function call and a callback procedure.

1) Write a callback procedure (PBCALLBACK):
You should write code to respond to the four PBCALLBACK codes:
PBC_OPEN, PBC_CLOSE, PBC_CANCEL, PBC_PROCESS
For the first 3 codes, TRUE signals success, FALSE signals failure.
PBC_PROCESS response codes are covered in detail in PROCBOX.H.

```
int CALLBACK _export PBCallback(HWND hwndPB, int iCode, LPARAM lParam)
{
        switch (iCode)
        {
                case PBC_OPEN:
                        // allocate memory, initialize variables
                return TRUE/FALSE;

                case PBC_CLOSE:
                        // deallocate memory
                return TRUE/FALSE;

                case PBC_CANCEL:
                        // maybe ask the user if he's sure he wants to cancel
                return TRUE/FALSE;

                case PBC_PROCESS:
                        // work on a chunk of the process here
                if (error)
                        return PBCR_ERROR;

                SendMessage(hwndPB, PM_SETGAUGE, <Percent_done>, 0l);
                return  PBCR_CONTINUE;
        }
```

```
        return FALSE;
}
```

2) In your code, call:

iError = **ProcessBox**(hwndOwner, lpszBoxTitle, lpfnPBCallback);

That's it!

(lpfnPBCallback is a procedure-instance address just like for dialog box procedures and other callbacks.)

Like DialogBox and MessageBox, ProcessBox returns after the box has been destroyed.

The Process Box API

More sophisticated uses of Process Box are possible with a few more (documented) APIs (see PROCBOX.H). For example, you can reuse the same process box with different callbacks or create several modeless Process Boxes which can run independently of each other. Here's the list of API functions split into three groups: beside each group is the name of a demo program which shows how to use them:

Single callback modal interface:(PROCAPP1.C)
(Run a Process Box and return)
ProcessBox - the simple API described above
ProcessBoxEx - the same, but adds bells and whistles

Multiple callback modal interface:(PROCAPP2.C)
(Attach different callbacks to the same Process Box)
CreateProcessBox - returns an HWND to a new ProcessBox
AttachProcess - attaches a callback to a ProcessBox
RunProcess - Runs a ProcessBox, returns when done/error/cancelled
DestroyProcessBox - like the name says

Low-level (modeless) interface:(PROCAPP3.C)
(Multiple simultaneously running modeless Proces Boxes)
IterateTaskProcesses - loops through the list of Process Boxes owned by the current task and send a PBC_PROCESS code
IsProcessMessage - the Process Box equivalent of IsDialogMessage

Lowest-level of access
The following functions are called for you by IterateTaskProcesses, but are available if you need them:
GetProcessBoxFirst - finds the first Process created by the current task
GetProcessBoxNext - used in conjunction with GetProcessBoxFirst

IterateProcess - when used in conjunction with GetProcessBox(First/Next) allows easy processing of active Process Boxes in a PeekMessage loop (or in response to a timer).

Custom process box interface: (PROCAPP4.C)
Although PROCBOX supplies a dialog box template, you can over-ride the standard Process Box and supply your own (with your own Dialog Box callback procedure). Like the default Process Box, Custom Process boxes can be used in Single callback modal, Multiple callback modal and Low-level (modeless) modes.

CustomProcessBox - like DialogBox()
CustomProcessBoxParam - like DialogBoxParam()
CreateCustomProcessBox - like CreateDialog()
CreateCustomProcessBoxParam - like CreateDialogParam()
SetCancelState - used in custom process dialog box procedure to indicate that process should be cancelled

The C-code and EXEs for the sample programs are provided. Make sure PROCBOX.DLL is in the same directory, the \Windows or \Windows\System directories. Click on the client area of the main windows or hit a key to start the demos.

To use the API functions, simply include PROCBOX.H in your file and add PROCBOX.LIB to your list of libraries.

Questions or comments?
Aneil Mallavarapu (mallav@itsa.ucsf.edu)