Launching Another Application under System 6

```
/*
 * Launching another application under System 6
 * From Apple Tech Note #126, example of launch/sublaunch code.
 * Use this code to launch or sublaunch applications under System 6.
 * Under System 7, you should use LaunchApplication or Apple Events
 * to launch an application.
 * NOTE:  If you are using THINK C you *must* call _exiting() before executing a
 * _Launch that will cause your program to quit.  The only case where you wouldn't
 * call _exiting() before a _Launch is when you sublaunch under
 * MultiFinder.  This code includes a 'test' for MultiFinder, although
 * you should make a version just for Finder or MF, if possible.
 * If you don't have ANSI in your project, you can link in _exiting() by adding the file
 * atexit.c to your project
 */

#define     APPLEID     128
#define     FILEID 129
#define     TRANSFER    1
#define     SUBLAUNCH   2
#define     QUIT        4

#include <stdlib.h>

MenuHandle myMenus[2];
Boolean done = false;

void Init(void);
OSErr DoLaunch(Boolean subLaunch);
void mainLoop (void);
void SetUpMenus (void);
Boolean mf_avail (void);
void DoCommand(long  mResult);

void
Init(void)
{
    InitGraf(&thePort);
    InitFonts();
    InitWindows();
    TEInit();
    InitDialogs(nil);
    InitCursor();
}

main()
{
    Init();
    SetUpMenus();
    mainLoop();
}

typedef struct LaunchStruct {
    unsigned char  *pfName;      /* pointer to the name of launchee */
```

```
        short           param;
        char            LC[2]; /* extended parameters: */
        long            extBlockLen;   /* number of bytes in extension == 6 */
        short           fFlags; /* Finder file info flags (see below) */
        long            launchFlags;    /* bit 31,30==1 for sublaunch,
                                         * others reserved */
} *pLaunchStruct;

pascal OSErr LaunchIt( pLaunchStruct pLnch) /* < 0 means error */
        = {0x205F, 0xA9F2, 0x3E80};

/* pops pointer into A0, calls Launch, pops D0 error code into result:
        MOVE.L  (A7)+,A0
        _Launch
        MOVE.W  D0,(A7)  ; since it MAY return */

OSErr DoLaunch(Boolean subLaunch)
                                /* Sublaunch if true and launch if false */
{       /* DoLaunch */
        struct LaunchStruct     myLaunch;
        Point                   where;          /* where to display dialog */
        SFReply                 reply;                  /* reply record */
        SFTypeList              myFileTypes;            /* we only want APPLs */
        short                   numFileTypes=1;
        HFileInfo               myPB;
        unsigned char           *dirNameStr;
        OSErr                   err;

        where.h = 80;
        where.v = 90;
        myFileTypes[0] = 'APPL';                /* we only want APPLs */
        /* Let the user choose the file to Launch */
        SFGetFile(where, "", nil, numFileTypes, myFileTypes, nil, &reply);

        if (reply.good)
        {
                dirNameStr = reply.fName;  /* initialize to file selected */

                /* Get the Finder flags */
                myPB.ioNamePtr = dirNameStr;
                myPB.ioVRefNum = reply.vRefNum;
                myPB.ioFDirIndex = 0;
                myPB.ioDirID = 0;
                err = PBGetCatInfo((CInfoPBPtr) &myPB,false);
                if (err != noErr)
                        return err;

                /* Set the current volume to where the target application is */
                err = SetVol(nil, reply.vRefNum);
                if (err != noErr)
                        return err;

                /* Set up the launch parameters */
                myLaunch.pfName = reply.fName;     /* pointer to our fileName */
                myLaunch.param = 0;             /* we don't want alternate screen
                                                 * or sound buffers */
```

```
            /* set up LC so as to tell Launch that there is non-junk next */
            myLaunch.LC[0] = 'L'; myLaunch.LC[1] = 'C';
            myLaunch.extBlockLen = 6;           /* length of param. block past
                                                 * this long word */
            /* copy flags; set bit 6 of low byte to 1 for RO access: */
            myLaunch.fFlags = myPB.ioFlFndrInfo.fdFlags;    /* from _GetCatInfo
                                                             */
            /* Test subLaunch and set launchFlags accordingly */
            if (subLaunch)
                  myLaunch.launchFlags = 0xC0000000;    /* set BOTH hi bits
                                                         * for a sublaunch
                                                         */
            else
                  myLaunch.launchFlags = 0x00000000;    /* Just launch then
                                                         * quit
                                                         */

            err = LaunchIt(&myLaunch);                  /* call _Launch */
            if (err < 0) {
            /* the launch failed, so put up an alert to inform the user */
                  return err;
            }
            else
                  return noErr;
      } /* if reply.good */
      else
            return -1;
} /* DoLaunch */

void mainLoop()
{
      WindowPtr     whichWindow;
      EventRecord   myEvent;
      Rect          dragRect = screenBits.bounds;
      char          theChar;
      short         partCode;

      do {
            if (WaitNextEvent(everyEvent, &myEvent, 0, nil)) {
                  switch (myEvent.what) {       /* case on event type */
                  case mouseDown:
                        partCode = FindWindow(myEvent.where,
                                    &whichWindow);
                        switch (partCode) {
                        case inSysWindow:
                              SystemClick(&myEvent, whichWindow);
                              break;
                        case inMenuBar:
                              DoCommand(MenuSelect(myEvent.where));
                              break;
                        }
                        break;
                  case keyDown:
                  case autoKey:   /* key pressed once or held down to repeat */
                        theChar = (myEvent.message & charCodeMask);
                                                /* get the char */
```

```
                        if (myEvent.modifiers & cmdKey)
                            DoCommand(MenuKey(theChar));
                        break;
                    }
                }
            } while (!done);
}

void SetUpMenus()
{
    myMenus[0] = NewMenu(APPLEID,"\p\024");
                                        /* that's shift-option-k (apple) */
    AppendMenu(myMenus[0],"\p(-");
    AddResMenu(myMenus[0], 'DRVR');
                        /* add desk accessory names to Apple menu */
    myMenus[1] = NewMenu(FILEID,"\pFile");
                        /* read file menu from resource file */
    AppendMenu(myMenus[1],"\p/T Transfer");
    AppendMenu(myMenus[1],"\p/S Sublaunch");
    AppendMenu(myMenus[1],"\p(-");
    AppendMenu(myMenus[1],"\p/Q Quit");
    InsertMenu(myMenus[0],0);
    InsertMenu(myMenus[1],0);

    DrawMenuBar();  /* and draw menu bar */
}

/*
 * mf_avail  -- is multifinder running or not? *
 * Supposedly _OSDispatch is only available under multifinder.
 */
#define OSDispatch      0xA88F
#define UnImplTrap      0xA09F

Boolean mf_avail()
{
    return  (NGetTrapAddress(OSDispatch, ToolTrap) !=
                NGetTrapAddress(UnImplTrap, ToolTrap));
}

/*
**  Execute menu command specified by mResult,
**  the result of MenuSelect
*/
void DoCommand(long mResult)
{
    short   theItem,        /* menu item number from mResult low-order word */
            theMenu;        /* menu number from mResult high-order word */
    Str255          name;   /* desk accessory name */
    short   temp;

    theItem = LoWord(mResult);      /* call Toolbox Utility routines to */
    theMenu = HiWord(mResult);      /* set menu item number and menu */

    switch (theMenu) {              /* switch on menu ID */
        case APPLEID:
```

```
                GetItem(myMenus[0], theItem, name);
                temp = OpenDeskAcc(name);
                break;
        case FILEID:
                switch (theItem) {
                case TRANSFER:
                        _exiting(1);            /* Very important! THINK C needs
                                                 * you to clean up */
                        DoLaunch(0); /* before a nonstandard exit like
                                                 *_Launch! */
                        break;
                case SUBLAUNCH:
                        if (!mf_avail())
                                _exiting(1);    /* If we're using Finder, we must
                                                 * clean up. */
                        DoLaunch(1);
                        break;
                case QUIT:
                        done = TRUE;
                        break;
                }
    }
    HiliteMenu(0);      /* Unhighlight menu title (highlighted by MenuSelect) */
}
```