```
;/* AppWindow.c - Execute me to compile me with Lattice 5.10a
lc -cfis -v -d0 -b1 -j73 AppWindow.c
Blink FROM LIB:c.o,AppWindow.o TO AppWindow LIBRARY LIB:LC.lib,LIB:Amiga.lib
quit
*/

#include <exec/memory.h>
#include <intuition/intuition.h>
#include <workbench/startup.h>
#include <workbench/workbench.h>

#ifdef LATTICE
#include <stdio.h>

/* disable SAS/C CTRL-C handing */
int
CXBRK(void)
{
    return (0);
}
int
chkabort(void)
{
    return (0);
}

#include <clib/exec_protos.h>
#include <clib/intuition_protos.h>
#include <clib/icon_protos.h>
#include <clib/wb_protos.h>
#endif

struct IntuitionBase *IntuitionBase;
struct WorkbenchBase *WorkbenchBase;

void            main(void);

void
main(void)
{
    struct MsgPort *msgport;
    struct Window  *window;
    struct AppWindow *appwindow;
    struct IntuiMessage *imsg;
    struct AppMessage *appmsg;
    struct WBArg   *argptr;

    ULONG           id = 1, userdata = 0;
    BOOL            ABORT = FALSE;
    UCOUNT          i;

    /* Open Intuition.library & Workbench.library. Fail silently if < 36 */
    if (IntuitionBase = OpenLibrary("intuition.library", 36))
    {
        if (WorkbenchBase = OpenLibrary("workbench.library", 36))
        {
            /* Create the message port to which Workbench can send messages */
            if (msgport = CreateMsgPort())
            {
                if (window =
                    OpenWindowTags(NULL, WA_Left, 0, WA_Top, 1, WA_Width, 160,
                                   WA_Height, 50, WA_IDCMP, CLOSEWINDOW,
                                   WA_Flags, WINDOWCLOSE | WINDOWDRAG,
                                   WA_Title, "AppWindow", TAG_END))
                {

                    /*
                     * Turn the window we opened into an AppWindow. Provide an
                     * ID so you can tell possible more AppWindows apart.
                     */
                    if (appwindow = AddAppWindow(id, userdata, window, msgport, NULL))
                    {
                        do
                        {
                            /* Wait for either a CLOSEWINDOW or an AppMessage */
                            Wait(1 << window->UserPort->mp_SigBit |
                                1 << msgport->mp_SigBit);
                            while (imsg = (struct IntuiMessage *)
                                GetMsg(window->UserPort))
                            {
                                if (imsg->Class = CLOSEWINDOW)
                                    ABORT = TRUE;
                                ReplyMsg((struct Message *) imsg);
                            }
                            while (appmsg = (struct AppMessage *) GetMsg(msgport))
                            {

                                /*
                                 * The AppMessage type will be MTYPE_APPWINDOW,
                                 * the ID & userdata are what we supplied when
                                 * the window was designed as an AppWindow.
                                 * NumArgs allows us to process the Workbench
                                 * arguments properly.
                                 */
                                printf(
                       "aw: appmsg=%lx, Type=%ld, ID=%ld, UserData=%ld, NumArgs=%ld\n",
                                       appmsg, appmsg->am_Type, appmsg->am_ID,
                                       appmsg->am_UserData, appmsg->am_NumArgs);

                                /*
                                 * Get a pointer to the start of the Workbench
                                 * argument list.
                                 */
                                argptr = appmsg->am_ArgList;
                                for (i = 0; i < appmsg->am_NumArgs; i++)
                                {

                                    /*
                                     * The lock will be on the directory in
                                     * which the file resides. If there is no
                                     * filename, either a volume or window was
                                     * dropped on us.
                                     */
                                    printf("\targ(%ld): Name='%s', Lock=%lx\n",
                                           i, argptr->wa_Name, argptr->wa_Lock);
                                    /* Point to next argument */
                                    argptr++;
                                }
                                ReplyMsg((struct Message *) appmsg);
                            }
                        } while (ABORT == FALSE);
                        /* remove the appwindow status and close down */
                        RemoveAppWindow(appwindow);
                    }
                    else
                        printf("Couldn't AddAppWindow\n");
                    CloseWindow(window);
                }
                else
                    printf("Couldn't open window\n");
                DeleteMsgPort(msgport);
            }
            else
                printf("Coulnd't create messageport\n");
            CloseLibrary(WorkbenchBase);
        }
        else
            printf("Couldn't open workbench.library\n");
        CloseLibrary(IntuitionBase);
    }
    else
        printf("Couldn't open intuition.library\n");
}
```

```
/*
 * AppIcon.h - Icon for AppIcon. Output from IconEd.
 *
 */

UWORD chip      AppIcon1IData[] =
{
/* Plane 0 */
    0x0000, 0x0000, 0x0000, 0x8000, 0x0000, 0x0000, 0x0001, 0x8000,
    0x0000, 0x0000, 0x0011, 0x8000, 0x0000, 0x0000, 0x0031, 0x8000,
    0x0000, 0x0000, 0x0231, 0x8000, 0x0000, 0x0000, 0x0631, 0x8000,
    0x0000, 0x0000, 0x4631, 0x8000, 0x0000, 0x0000, 0xC631, 0x8000,
    0x0000, 0x0000, 0xC631, 0x8000, 0x0000, 0x0000, 0xC631, 0x8000,
    0x0000, 0x0000, 0xC631, 0x8000, 0x0000, 0x0000, 0xC631, 0x8000,
    0x0000, 0x0000, 0xC631, 0x8000, 0x0000, 0xFFFF, 0xC631, 0x8000,
    0x0000, 0x0000, 0x0631, 0x8000, 0x001F, 0xFFFF, 0xFE31, 0x8000,
    0x0000, 0x0000, 0x0031, 0x8000, 0x03FF, 0xFFFF, 0xFFF1, 0x8000,
    0x0000, 0x0000, 0x0001, 0x8000, 0x7FFF, 0xFFFF, 0xFFFF, 0x8000,
/* Plane 1 */
    0xFFFF, 0xFFFF, 0xFFFF, 0x0000, 0xC000, 0x0000, 0x0000, 0x0000,
    0xC7FF, 0xFFFF, 0xFFE0, 0x0000, 0xC600, 0x0000, 0x0000, 0x0000,
    0xC63F, 0xFFFF, 0xFC00, 0x0000, 0xC630, 0x0000, 0x0000, 0x0000,
    0xC631, 0xFFFF, 0x8000, 0x0000, 0xC631, 0x8000, 0x0000, 0x0000,
    0xC631, 0x8000, 0x0000, 0x0000, 0xC631, 0x8000, 0x0000, 0x0000,
    0xC631, 0x8000, 0x0000, 0x0000, 0xC631, 0x8000, 0x0000, 0x0000,
    0xC631, 0x8000, 0x0000, 0x0000, 0xC631, 0x0000, 0x0000, 0x0000,
    0xC630, 0x0000, 0x0000, 0x0000, 0xC620, 0x0000, 0x0000, 0x0000,
    0xC600, 0x0000, 0x0000, 0x0000, 0xC400, 0x0000, 0x0000, 0x0000,
    0xC000, 0x0000, 0x0000, 0x0000, 0x8000, 0x0000, 0x0000, 0x0000,
};

struct Image    AppIconI1 =
{
    0, 0,                                       /* Upper left corner */
    49, 20, 2,                                  /* Width, Height, Depth */
    AppIcon1IData,                              /* Image data */
    0x0003, 0x0000,                             /* PlanePick, PlaneOnOff */
    NULL                                        /* Next image */
};

UWORD chip      AppIconI2Data[] =
{
/* Plane 0 */
    0xFFFF, 0xFFFF, 0xFFFF, 0x0000, 0xC000, 0x0000, 0x0000, 0x0000,
    0xC7FF, 0xFFFF, 0xFFE0, 0x0000, 0xC600, 0x0000, 0x0000, 0x0000,
    0xC63F, 0xFFFF, 0xFC00, 0x0000, 0xC630, 0x0000, 0x0000, 0x0000,
    0xC631, 0xFFFF, 0x8000, 0x0000, 0xC631, 0x8000, 0x0000, 0x0000,
    0xC631, 0x8000, 0x0000, 0x0000, 0xC631, 0x8000, 0x0000, 0x0000,
    0xC631, 0x8000, 0x0000, 0x0000, 0xC631, 0x8000, 0x0000, 0x0000,
    0xC631, 0x8000, 0x0000, 0x0000, 0xC631, 0x0000, 0x0000, 0x0000,
    0xC630, 0x0000, 0x0000, 0x0000, 0xC620, 0x0000, 0x0000, 0x0000,
    0xC600, 0x0000, 0x0000, 0x0000, 0xC400, 0x0000, 0x0000, 0x0000,
    0xC000, 0x0000, 0x0000, 0x0000, 0x8000, 0x0000, 0x0000, 0x0000,
/* Plane 1 */
    0x0000, 0x0000, 0x0000, 0x8000, 0x0000, 0x0000, 0x0001, 0x8000,
    0x0000, 0x0000, 0x0011, 0x8000, 0x0000, 0x0000, 0x0031, 0x8000,
    0x0000, 0x0000, 0x0231, 0x8000, 0x0000, 0x0000, 0x0631, 0x8000,
    0x0000, 0x0000, 0x4631, 0x8000, 0x0000, 0x0000, 0xC631, 0x8000,
    0x0000, 0x0000, 0xC631, 0x8000, 0x0000, 0x0000, 0xC631, 0x8000,
    0x0000, 0x0000, 0xC631, 0x8000, 0x0000, 0x0000, 0xC631, 0x8000,
    0x0000, 0x0000, 0xC631, 0x8000, 0x0000, 0xFFFF, 0xC631, 0x8000,
    0x0000, 0x0000, 0x0631, 0x8000, 0x001F, 0xFFFF, 0xFE31, 0x8000,
    0x0000, 0x0000, 0x0031, 0x8000, 0x03FF, 0xFFFF, 0xFFF1, 0x8000,
    0x0000, 0x0000, 0x0001, 0x8000, 0x7FFF, 0xFFFF, 0xFFFF, 0x8000,
};

struct Image    AppIconI2 =
{
    0, 0,                                       /* Upper left corner */
    49, 20, 2,                                  /* Width, Height, Depth */
    AppIconI2Data,                              /* Image data */
    0x0003, 0x0000,                             /* PlanePick, PlaneOnOff */
    NULL                                        /* Next image */
};

struct DiskObject AppIconDObj =
{
    NULL,                                       /* Magic Number */
    NULL,                                       /* Version */
    {                                           /* Embedded Gadget Structure */
        NULL,                                   /* Next Gadget Pointer */
        0, 0, 49, 21,                           /* Left,Top,Width,Height */
        GADGHIMAGE,                             /* Flags */
        NULL,                                   /* Activation Flags */
        NULL,                                   /* Gadget Type */
        (APTR) & AppIconI1,                     /* Render Image */
        (APTR) & AppIconI2,                     /* Select Image */
        NULL,                                   /* Gadget Text */
        NULL,                                   /* Mutual Exclude */
        NULL,                                   /* Special Info */
        0,                                      /* Gadget ID */
        NULL,                                   /* User Data */
    },
    NULL,                                       /* Icon Type */
    NULL,                                       /* Default Tool */
    NULL,                                       /* Tool Type Array */
    NO_ICON_POSITION,                           /* Current X */
    NO_ICON_POSITION,                           /* Current Y */
    NULL,                                       /* Drawer Structure */
    NULL,                                       /* Tool Window */
    NULL                                        /* Stack Size */
};
```

```
;/* AppIcon.c - Execute me to compile me with Lattice 5.10a
lc -cfis -v -d0 -b1 -j73 AppIcon.c
Blink FROM LIB:c.o,AppIcon.o TO AppIcon LIBRARY LIB:LC.lib,LIB:Amiga.lib
quit
*/

#include <intuition/intuition.h>
#include <exec/memory.h>
#include <workbench/startup.h>
#include <workbench/workbench.h>

#include "appicon.h"

#ifdef LATTICE
#include <stdio.h>

/* disable SAS/C CTRL-C handing */
int
CXBRK(void)
{
    return (0);
}
int
chkabort(void)
{
    return (0);
}

#include <clib/exec_protos.h>
#include <clib/intuition_protos.h>
#include <clib/wb_protos.h>
#endif

struct IntuitionBase *IntuitionBase;
struct WorkbenchBase *WorkbenchBase;

void            main(void);

void
main(void)
{
    struct MsgPort *msgport;
    struct Window  *window;
    struct AppIcon *appicon;
    struct IntuiMessage *imsg;
    struct AppMessage *appmsg;
    struct WBArg   *argptr;

    ULONG           id = 1, userdata = 0;
    BOOL            ABORT = FALSE;
    UCOUNT          i;


    /* Open needed libraries. Fail silently if < 36 */
    if (IntuitionBase = OpenLibrary("intuition.library", 36))
    {
        if (WorkbenchBase = OpenLibrary("workbench.library", 36))
        {
            if (msgport = CreateMsgPort())
            {
                if (window =
                    OpenWindowTags(NULL, WA_Left, 0, WA_Top, 1, WA_Width, 160,
                                   WA_Height, 50, WA_IDCMP, CLOSEWINDOW,
                                   WA_Flags, WINDOWCLOSE | WINDOWDRAG,
                                   WA_Title, "AppIcon", TAG_END))
                {

                    /* Add the icon to Workbench */
                    if (appicon = AddAppIcon(id, userdata, "AppIcon",
                        msgport, NULL, &AppIconDObj, NULL))
                    {
                        do
                        {
                            Wait(1 << window->UserPort->mp_SigBit |
                                1 << msgport->mp_SigBit);
                            while (imsg = (struct IntuiMessage *)
                                GetMsg(window->UserPort))
                            {
                                if (imsg->Class = CLOSEWINDOW)
                                    ABORT = TRUE;
                                ReplyMsg((struct Message *) imsg);
                            }
                            while (appmsg = (struct AppMessage *) GetMsg(msgport))
                            {
                                printf(
                        "ai: appmsg=%lx, Type=%ld, ID=%ld, UserData=%ld, NumArgs=%ld\n",
                                       appmsg, appmsg->am_Type, appmsg->am_ID,
                                       appmsg->am_UserData, appmsg->am_NumArgs);
                                argptr = appmsg->am_ArgList;

                                /*
                                 * If am->NumArgs is zero the user
                                 * double-clicked on our icon, otherwise one or
                                 * more icons were dropped on top of it.
                                 */
                                for (i = 0; i < appmsg->am_NumArgs; i++)
                                {
                                    printf("\targ(%ld): Name='%s', Lock=%lx\n",
                                    i, argptr->wa_Name, argptr->wa_Lock);
                                    argptr++;
                                }
                                ReplyMsg((struct Message *) appmsg);
                            }
                        } while (ABORT == FALSE);
                        /* Remove the AppIcon and clean up */
                        RemoveAppIcon(appicon);
                    }
                    else
                        printf("Couldn't add AppIcon\n");
                    CloseWindow(window);
                }
                else
                    printf("Couldn't open window\n");
                DeleteMsgPort(msgport);
            }
            else
                printf("Couldn't create messageport\n");
            CloseLibrary(WorkbenchBase);
        }
        else
            printf("Couldn't open workbench.library\n");
        CloseLibrary(IntuitionBase);
    }
    else
        printf("Couldn't open intuition.library\n");
}
```

```
;/* AppMenu.c - Execute me to compile me with Lattice 5.10a
lc -cfis -v -d0 -b1 -j73 AppMenu.c
Blink FROM LIB:c.o,AppMenu.o TO AppMenu LIBRARY LIB:LC.lib,LIB:Amiga.lib
quit
*/

#include <intuition/intuition.h>
#include <exec/memory.h>
#include <workbench/startup.h>
#include <workbench/workbench.h>

#ifdef LATTICE

/* disable SAS/C CTRL-C handing */
int
CXBRK(void)
{
    return (0);
}
int
chkabort(void)
{
    return (0);
}

#include <clib/exec_protos.h>
#include <clib/intuition_protos.h>
#include <clib/icon_protos.h>
#include <clib/wb_protos.h>
#include <clib/dos_protos.h>
#include <clib/alib_stdio_protos.h>
#endif

struct IntuitionBase *IntuitionBase;
struct WorkbenchBase *WorkbenchBase;

void            main(void);

void
main(void)
{
    struct MsgPort *msgport;
    struct Window  *window;
    struct AppMenuItem *appmenuitem;
    struct IntuiMessage *imsg;
    struct AppMessage *appmsg;
    struct WBArg   *argptr;

    ULONG           id = 1, userdata = 0, i;
    BOOL            ABORT = FALSE;

    /* Open Intuition.library & Workbench.library. Fail silently if < 36 */
    if (IntuitionBase = OpenLibrary("intuition.library", 36))
    {
        if (WorkbenchBase = OpenLibrary("workbench.library", 36))
        {
            /* Create the message port to which Workbench can send messages */
            if (msgport = CreateMsgPort())
            {
                if (window =
                    OpenWindowTags(NULL, WA_Left, 0, WA_Top, 1, WA_Width, 160,
                                   WA_Height, 50, WA_IDCMP, CLOSEWINDOW,
                                   WA_Flags, WINDOWCLOSE | WINDOWDRAG,
                                   WA_Title, "AppMenu", TAG_END))
                {

                    /* Use our window to attach an menu item to the Tools menu. */
                    if (appmenuitem = AddAppMenuItem(id, userdata,
                                                     "AppMenuItem", msgport, NULL))
                    {

                        do
                        {
                            /* Wait for either a CLOSEWINDOW or an AppMessage */
                            Wait(1 << window->UserPort->mp_SigBit |
                                 1 << msgport->mp_SigBit);
                            while (imsg = (struct IntuiMessage *)
                                GetMsg(window->UserPort))
                            {
                                if (imsg->Class = CLOSEWINDOW)
                                    ABORT = TRUE;
                                ReplyMsg((struct Message *) imsg);
                            }
                            while (appmsg = (struct AppMessage *) GetMsg(msgport))
                            {

                                /*
                                 * The AppMessage type will be MTYPE_APPMENU,
                                 * the ID & userdata are what we supplied when
                                 * the window was designed as an AppWindow.
                                 * Since there are no Workbench arguments for
                                 * menu operations, NumArgs will always be 0.
                                 */
                                printf(
                       "am: appmsg=%lx, Type=%ld, ID=%ld, UserData=%ld, NumArgs=%ld\n",
                                       appmsg, appmsg->am_Type, appmsg->am_ID,
                                       appmsg->am_UserData, appmsg->am_NumArgs);
                                argptr = appmsg->am_ArgList;
                                for (i = 0; i < appmsg->am_NumArgs; i++)
                                {

                                    /*
                                     * The lock will be on the directory in
                                     * which the file resides. If there is no
                                     * filename, either a volume or window was
                                     * dropped on us.
                                     */
                                    printf("\targ(%ld): Name='%s', Lock=%lx\n", i,
                                           argptr->wa_Name, argptr->wa_Lock);
                                    /* Point to next argument */
                                    argptr++;
                                }

                                ReplyMsg((struct Message *) appmsg);
                            }
                        } while (ABORT == FALSE);
                        /* remove the AppMenu and close down */
                        RemoveAppMenuItem(appmenuitem);
                    }
                    else
                        printf("Couldn't add AppMenuItem\n");
                    CloseWindow(window);
                }
                else
                    printf("Couldn't open window\n");
                DeleteMsgPort(msgport);
            }
            else
                printf("Coulnd't create messageport\n");
            CloseLibrary(WorkbenchBase);
        }
        else
            printf("Couldn't open workbench.library\n");
        CloseLibrary(IntuitionBase);
    }
    else
        printf("Couldn't open intuition.library\n");
}
```