


```

{
    /* Application code body...*/

    /* Put up file requester */
    if (AslRequest(fr, 0L))
    {
        /* If the file requester's rf_NumArgs field
        ** is not zero, the user multiselect. The
        ** number of files is stored in rf_NumArgs.
        */
        if (fr->rf_NumArgs)
        {
            /* rf_ArgList is an array of WBArg structures
            ** (defined in <workbench/startup.h>).
            ** Each entry in the WBArg array corresponds
            ** to one of the files the user selected
            ** (the entries are in alphabetical order).
            */
            wargs = fr->rf_ArgList;

            /* The user multiselect, step through
            ** the list of selected files.
            */
            for (x=0; x < fr->rf_NumArgs; x++)
                printf("Argument %d - %s\n", x,
                    fr->rf_Dir, wargs[x].wa_Name);
        }
        else
            /* The user didn't multiselect, use the
            ** normal way to get the file name.
            */
            printf("%s\n", fr->rf_Dir, fr->rf_File);
    }
    /* More application code body... */

    /* Done with the FileRequester, better return it */
    FreeAslRequest(fr);

    CloseWindow(window);
}
CloseScreen(screen);
}
CloseLibrary(IntuitionBase);
}
CloseLibrary(AslBase);
}
}

```

```

/* fontreq.c - Execute me to compile me with Lattice 5.10
LC -bl -cfistq -v -y -j73 fontreq.c
Blink FROM LIB:c.o,fontreq.o TO fontreq LIBRARY LIB:LC.lib,LIB:Amiga.lib
quit
*/

#include <clib/asl_protos.h>
#include <clib/exec_protos.h>
#include <clib/alib_stdio_protos.h>
#include <exec/libraries.h>

#ifdef LATTICE
int CXBRK(void) { return(0); } /* Disable Lattice CTRL/C handling */
int chkabort(void) { return(0); } /* really */
#endif

UBYTE *vers = "\0$VER: fontreq 1.0";

void main(void);
struct Library *AslBase, *UtilityBase;

/* The replacement strings for the "mode" cycle gadget. The
** first string is the cycle gadget's label. The other strings
** are the actual strings that will appear on the cycle gadget.
*/
UBYTE *modelist[] =
{
    "RKM Modes",
    "Mode 0",
    "Mode 1",
    "Mode 2",
    "Mode 3",
    "Mode 4",
    "Mode 5",
    "Mode 6",
    "Mode 7",
    "Mode 8",
    "Mode 9",
    NULL
};

void main()
{
    struct FontRequester *fr;

    if (AslBase = OpenLibrary("asl.library", 36L))
    {
        if (fr = (struct FontRequester *)
            AllocAslRequestTags(ASL_FontRequest,
                /* tell the requester to use my custom mode names */
                ASL_ModeList, modelist,

                /* Supply initial values for requester */
                ASL_FontName, (ULONG)"topaz.font",
                ASL_FontHeight, 11L,
                ASL_FontStyles, FSF_BOLD | FSF_ITALIC,
                ASL_FrontPen, 0x00L,
                ASL_BackPen, 0x01L,

                /* Only display font sizes between 8 and
                ** 14, inclusive. */
                ASL_MinHeight, 8L,
                ASL_MaxHeight, 14L,

                /* Give us all the gadgetry, but only display
                ** fixed width fonts */
                ASL_FuncFlags, FONF_FRONTCOLOR | FONF_BACKCOLOR |
                    FONF_DRAWMODE | FONF_STYLES | FONF_FIXEDWIDTH,
                TAG_DONE))
        {
            /* application code here... */

```

```

/* Pop up the requester */
if (AslRequest(fr, 0L))
{
    /* The user selected something, report their choice */
    printf("%s\n ySize = %d Style = 0x%x Flags = 0x%x\n",
        " Ppen = 0x%x Bpen = 0x%x DrawMode = 0x%x\n",
        fr->fo.Attr.ta_Name,
        fr->fo.Attr.ta_YSize,
        fr->fo.Attr.ta_Style,
        fr->fo.Attr.ta_Flags,
        fr->fo_PromptPen,
        fr->fo_BackPen,
        fr->fo_DrawMode);
}
else
/* The user cancelled the requester, or
** some kind of error occurred preventing
** the requester from opening. */
printf("Request Cancelled\n");

/* more application code here ...*/

FreeAslRequest(fr);
}
CloseLibrary(AslBase);
}

```

```

/* Filehook.c - Execute me to compile me with lattice 5.10
LC -bl -cfisq -v -y -j73 filehook.c
Blink FROM LIB:c.o,filehook.o TO filehook LIBRARY LIB:LC.lib,LIB:Amiga.lib
quit
*/
#include <clib/asl_protos.h>
#include <clib/exec_protos.h>
#include <clib/dos_protos.h>
#include <clib/intuition_protos.h>
#include <clib/alib_stdio_protos.h>
#include <dos/dosasl.h>
#include <intuition/intuition.h>
#include <exec/libraries.h>

#ifdef LATTICE
int CxBRK(void) { return(0); } /* Disable Lattice CTRL/C handling */
int chxabort(void) { return(0); } /* really */
#endif

#define DESTPATLENGTH 20

UBYTE *vers = "\$VER: filehook 1.0";

void main(void)
{
    struct Library *AslBase;
    struct IntuitionBase *IntuitionBase;
    struct Window *Window;

    CPTR HookFunc();

    /* this is the pattern matching string that the hook function uses */
    UBYTE *sourcepattern = "(#?.info)";
    UBYTE pat[DESTPATLENGTH];

    void main()
    {
        struct FileRequester *fr;

        /* This is a dos.library function that turns a pattern matching
        ** string into something the DOS pattern matching functions
        ** can understand.
        */
        ParsePattern(sourcepattern, pat, DESTPATLENGTH);

        if (AslBase = OpenLibrary("asl.library", 36L))
        {
            if (IntuitionBase = (struct IntuitionBase *)
                OpenLibrary("intuition.library", 36L))
            {
                /* Open a window that gets ACTIVEWINDON events */
                if (Window = (struct Window *)OpenWindowTags(NULL,
                    WA_Title, "ASL Hook Function Example",
                    WA_IDCMP, IDCMP_ACTIVIEWINDOW,
                    WA_Flags, WINDOWDEPTH,
                    TAG_END))
                {
                    if (fr = AllocFileRequest())
                    {
                        /* application body here... */

                        if (AslRequestTags(fr,
                            ASL_Window, window,
                            ASL_TopEdge, 0L,
                            ASL_Height, 200L,
                            ASL_Hall, (ULONG)"Pick an icon to save",
                            ASL_HookFunc, (ULONG)HookFunc,
                            ASL_FuncFlags, FILE_DOWIDPFUNC | FILE_DOSGPFUNC | FILE_SAVE,
                            ASL_OKText, (ULONG)"Save",
                            TAG_DONE))
                        {
                            printf("You picked %s\n", fr->rF_Dir, fr->rF_File);
                        }
                    }
                }
            }
        }
    }
}

```

```

        /* more application body here */
        FreeFileRequest(fr);
    }
    CloseWindow(window);
}
CloseLibrary(IntuitionBase);
}
CloseLibrary(AslBase);
}

CPTR HookFunc(LONG type, CPTR obj, struct FileRequester *fr)
{
    static BOOL returnvalue;
    switch(type)
    {
        case FILE_DOMSGFUNC:
            /* We got a message meant for the window */
            printf("You activated the window\n");
            return(obj);
            break;
        case FILE_DOWILDFUNC:
            /* We got an AnchorPath structure, should
            ** the requester display this file? */

            /* MatchPattern() is a dos.library function that
            ** compares a matching pattern (parsed by the
            ** ParsePattern() DOS function) to a string and
            ** returns true if they match. */
            returnvalue = MatchPattern(pat,
                ((struct AnchorPath *)obj)->ap_Info.fib_FileName);

            /* we have to negate MatchPattern()'s return value
            ** because the file requester expects a zero for
            ** a match not a TRUE value */
            return( (CPTR)(! returnvalue) );
            break;
    }
}

```

