```
;/* simpleFR.c - Execute me to compile me with SASC 5.10
LC -b1 -cfistq -v -y -j73 simpleFR.c
Blink FROM LIB:c.o,simpleFR.o TO simpleFR 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 <dos/dosasl.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: simpleFR 1.1";

void main(void);

#define MYLEFTEDGE 0
#define MYTOPEDGE  0
#define MYWIDTH    320
#define MYHEIGHT   400

struct Library *AslBase;

struct TagItem frtags[] =
{
    ASL_Hail,       (ULONG)"The RKM file requester",
    ASL_Height,     MYHEIGHT,
    ASL_Width,      MYWIDTH,
    ASL_LeftEdge,   MYLEFTEDGE,
    ASL_TopEdge,    MYTOPEDGE,
    ASL_OKText,     (ULONG)"O KAY",
    ASL_CancelText, (ULONG)"not OK",
    ASL_File,       (ULONG)"asl.library",
    ASL_Dir,        (ULONG)"libs:",
    TAG_DONE
};


void main()
{

    struct FileRequester *fr;

    if (AslBase = OpenLibrary("asl.library", 36L))
    {
        if (fr = (struct FileRequester *)
            AllocAslRequest(ASL_FileRequest, frtags))
        {
            /* Application body, blah, blah,... */

            /* Application need a requester */
            if (AslRequest(fr, 0L))
                printf("file choice = %s%s\n", fr->rf_Dir, fr->rf_File);
            else
                printf("User Cancelled\n");

            /* More application body, blah, blah... */
        }

        /* Don't need any more requesters, better
        ** give the requester structure back.
        */
        FreeAslRequest(fr);
    }
    CloseLibrary(AslBase);
}
```

```
;/* filepat.c - Execute me to compile me with SASC 5.10
LC -b1 -cfistq -v -y -j73 filepat.c
Blink FROM LIB:c.o,filepat.o TO filepat LIBRARY LIB:LC.lib,LIB:Amiga.lib
quit
*/

#include <clib/asl_protos.h>
#include <clib/exec_protos.h>
#include <clib/intuition_protos.h>
#include <clib/alib_stdio_protos.h>
#include <workbench/startup.h>
#include <intuition/intuition.h>
#include <intuition/screens.h>
#include <graphics/displayinfo.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: filepat 1.0";

void main(void);

struct Library *AslBase;
struct IntuitionBase *IntuitionBase;
struct Screen *screen;
struct Window *window;

struct WBArg *wbargs;

LONG x;


void main()
{

    struct FileRequester *fr;

    if (AslBase = OpenLibrary("asl.library", 36L))
    {
        if (IntuitionBase = (struct IntuitionBase *)
                OpenLibrary("intuition.library", 36L))
        {
            if (screen = (struct Screen *)OpenScreenTags(NULL,
                    SA_DisplayID, HIRESLACE_KEY,
                    SA_Title, "ASL Test Screen",
                    TAG_END))
            {
                if (window = (struct Window *)OpenWindowTags(NULL,
                        WA_CustomScreen, screen,
                        WA_Title, "ASL Test Window",
                        WA_Flags, WINDOWDEPTH | WINDOWDRAG,
                        TAG_END))
                {
                    if (fr = (struct FileRequester *)
                        AllocAslRequestTags(ASL_FileRequest,
                            ASL_Hail, (ULONG)"RKM File Requester, FilePat",
                            ASL_Dir,  (ULONG)"libs:",
                            ASL_File, (ULONG)"asl.library",

                            /* The initial pattern string for the
                            ** pattern gadget.
                            */
                            ASL_Pattern, (ULONG)"~(rexx#?|math#?)",

                            /* turn on multiselection and the pattern
                            ** matching gadget.
                            */
                            ASL_FuncFlags, FILF_MULTISELECT | FILF_PATGAD,

                            /* This requester is associated with this
                            ** window (and uses its message port).
                            */
                            ASL_Window, window,
                            TAG_DONE))
```

```
                    {
                        /* Application code body...*/

                        /* Put up file requester */
                        if (AslRequest(fr, 0L))
                        {
                            /* If the file requester's rf_NumArgs field
                            ** is not zero, the user multiselected. 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).
                                */
                                wbargs = fr->rf_ArgList;

                                /* The user multiselected, step through
                                ** the list of selected files.
                                */
                                for ( x=0;  x < fr->rf_NumArgs;  x++ )
                                    printf("Argument %d - %s%s\n", x,
                                        fr->rf_Dir, wbargs[x].wa_Name);
                            }
                            else
                                /* The user didn't multiselect, use the
                                ** normal way to get the file name.
                                */
                                printf("%s%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 -b1 -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"
                       "  FPen = 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_FrontPen,
                               fr->fo_BackPen,
                               fr->fo_DrawMode);
            }
            else
                /* The user cancelled the requester, or
                ** some kind of error occured 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 -b1 -cfistq -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 chkabort(void) { return(0); }  /* really */
#endif

#define DESTPATLENGTH 20

UBYTE *vers = "\0$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 ACTIVEWINDOW events */
            if (window = (struct Window *)OpenWindowTags(NULL,
                    WA_Title, "ASL Hook Function Example",
                    WA_IDCMP, IDCMP_ACTIVEWINDOW,
                    WA_Flags, WINDOWDEPTH,
                    TAG_END))
            {
                if (fr = AllocFileRequest())
                {
                    /* application body here... */


                    if (AslRequestTags(fr,
                        ASL_Window, window,
                        ASL_TopEdge, 0L,
                        ASL_Height, 200L,
                        ASL_Hail, (ULONG)"Pick an icon to save",
                        ASL_HookFunc, (ULONG)HookFunc,
                        ASL_FuncFlags, FILF_DOWILDFUNC | FILF_DOMSGFUNC | FILF_SAVE,
                        ASL_OKText, (ULONG)"Save",
                        TAG_DONE))
                    {
                        printf("You picked %s%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 FILF_DOMSGFUNC:
        /* We got a message meant for the window */
            printf("You activated the window\n");
            return(obj);
            break;
        case FILF_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;
    }
}
```

❖