```
;/* SignalNotification.c - Compiled with SAS/C 5.10a
lc -cfis -v -d0 -b0 -j73 SignalNotification.c
blink from SignalNotification.o to SignalNotification lib lib:amiga.lib
quit
*/

#include <exec/types.h>
#include <exec/memory.h>
#include <dos/dos.h>
#include <dos/dosasl.h>
#include <dos/notify.h>
#include <dos/rdargs.h>

#include <clib/exec_protos.h>
#include <clib/dos_protos.h>

/* Use pragmas if you've got them */

static UBYTE   *VersTag = "\0$VER: SignalNotification 37.1 (09.07.91)";
struct Library *DOSBase;
struct ExecBase *SysBase;

VOID            main(VOID);

VOID
main(VOID)
{

    struct RDArgs  *readargs;
    LONG            rargs[2];
    struct NotifyRequest *notifyrequest;
    UBYTE          *filename;
    ULONG           signr, signal;

    /* To appease amiga.lib */
    SysBase = (*((struct Library **) 4));

    /* Fail silently if < 37 */
    if (DOSBase = OpenLibrary("dos.library", 37))
    {
        /* See the DOS Autodocs for more information about ReadArgs() */
        if (readargs = ReadArgs("Filename/A", rargs, NULL))
        {
            filename = (UBYTE *) (rargs[0]);

            /* Allocate a NotifyRequest structure */
            if (notifyrequest = AllocMem(sizeof(struct NotifyRequest), MEMF_CLEAR))
            {
                /* And allocate a signalbit */
                if ((signr = AllocSignal(-1L)) != -1)
                {

                    /* Initialize notifcation request */
                    notifyrequest->nr_Name = filename;
                    notifyrequest->nr_Flags = NRF_SEND_SIGNAL | NRF_NOTIFY_INITIAL;
                    /* Signal this task */
                    notifyrequest->nr_stuff.nr_Signal.nr_Task =
                            (struct Task *) FindTask(NULL);
                    /* with this signal bit */
                    notifyrequest->nr_stuff.nr_Signal.nr_SignalNum = signr;

                    if ((StartNotify(notifyrequest)) == DOSTRUE)
                    {
                        /* Loop until Ctrl-C to exit */
                        for (;;)
                        {
                            signal = Wait(1L << signr | SIGBREAKF_CTRL_C);
                            if (signal & (1L << signr))
                                VFPrintf(Output(), "Notification signal!\n", NULL);
                            if (signal & SIGBREAKF_CTRL_C)
                            {
                                EndNotify(notifyrequest);
                                PrintFault(ERROR_BREAK, NULL);
                                break;
                            }
                        }
                    }
                    else
                        PrintFault(ERROR_NOT_IMPLEMENTED, NULL);    /* most logical */

                    FreeSignal(signr);
                }
                else
                    VFPrintf(Output(), "No signal available\n", NULL);
                FreeMem(notifyrequest, sizeof(struct NotifyRequest));
            }
            else
                PrintFault(ERROR_NO_FREE_STORE, NULL);

            FreeArgs(readargs);
        }
        else
            PrintFault(IoErr(), NULL);
        CloseLibrary(DOSBase);
    }
}
```

```
;/* MessageNotification.c - Compiled with SAS/C 5.10a:
lc -cfis -v -d0 -b0 -j73 MessageNotification.c
blink from MessageNotification.o to MessageNotification lib lib:amiga.lib
quit
*/

#include <exec/types.h>
#include <exec/memory.h>
#include <exec/lists.h>
#include <exec/nodes.h>
#include <dos/dos.h>
#include <dos/dosasl.h>
#include <dos/notify.h>
#include <dos/rdargs.h>

#include <clib/alib_protos.h>
#include <clib/exec_protos.h>
#include <clib/dos_protos.h>

/* Use pragmas if you've got them */

static UBYTE   *VersTag = "\0$VER: MessageNotification 37.1 (09.07.91)";
struct Library *DOSBase;
struct ExecBase *SysBase;

/* I'll keep track of the NotifyRequests in an Exec list */
struct NotifyNode
{
    struct Node     nn_Node;
    struct NotifyRequest nn_NotifyRequest;
    /* and whatever is useful. Maybe a FileInfoBlock structure... */
};

VOID            main(VOID);

VOID
main(VOID)
{

    struct RDArgs  *readargs;
    LONG            rargs[2];
    struct NotifyRequest *notifyrequest;
    struct NotifyMessage *notifymsg;
    struct List    *notifylist;
    struct MsgPort *notifyport;
    struct NotifyNode *nnode, *nextnode;
    UBYTE         **filenames;
    ULONG           signal, notifysignal;
    /* To appease amiga.lib */
    SysBase = (*((struct Library **) 4));

    /* Fail silently if < 37 */
    if (DOSBase = OpenLibrary("dos.library", 37))
    {

        /* See the DOS Autodocs for more information about ReadArgs() */
        if (readargs = ReadArgs("Filename/A/M", rargs, NULL))
        {
            /* Pointer to array of filenames */
            filenames = (UBYTE **) (rargs[0]);

            if (notifyport = CreateMsgPort())
            {
                if (notifylist = AllocMem(sizeof(struct List), MEMF_CLEAR))
                {
                    /* initialize list */
                    NewList(notifylist);

                    /* The list of filenames is terminated with a NULL */
                    while (*filenames)
                    {
                        /* Get a NotifyNode */
                        if (nnode = AllocMem(sizeof(struct NotifyNode), MEMF_CLEAR))
                        {
                            /*
                             * Use ln_Name to store qualified filename to
                             * monitor. Note that I keep using that pointer to
                             * the command argument line here. In a real life
                             * application you’re better off copying the
                             * string.
                             */
                            nnode->nn_Node.ln_Name = (UBYTE *) * filenames++;
                            notifyrequest = &(nnode->nn_NotifyRequest);

                            /* Initialize notifcation request */
                            notifyrequest->nr_Name = nnode->nn_Node.ln_Name;
                            notifyrequest->nr_Flags =
                                NRF_SEND_MESSAGE | NRF_WAIT_REPLY;
                            notifyrequest->nr_stuff.nr_Msg.nr_Port = notifyport;
                            /*
                             * I’m storing the address of the NotifyNode in
                             * nr_UserData. Not going to do anything with it
                             * here, but it would enable me to use the node
                             * immediately when I receive a NotifyMessage.
                             */
                            notifyrequest->nr_UserData = (ULONG) nnode;
                            /*
                             * only add the node to the list if notification is
                             * supported
                             */
                            if ((StartNotify(notifyrequest)) == DOSTRUE)
                            {
                                AddTail(notifylist, (struct Node *) nnode);
                                Printf("Notification on %s\n",
                                       (LONG) nnode->nn_Node.ln_Name);
                            }
                            else
                            {
                                Printf("Notification failed on %s\n",
                                       (LONG) nnode->nn_Node.ln_Name);
                                FreeMem(nnode, sizeof(struct NotifyNode));
                            }
                        }
                        else
                        {
                            PrintFault(ERROR_NO_FREE_STORE, NULL);
                            break;
                        }
                    }

                    /*
                     * Is list empty? If so get out of here. (Macro defined in
                     * <exec/lists.h>)
                     */
                    if (!(IsListEmpty(notifylist)))
                    {
                        /*
                         * No empty, so we’ve got outstanding NotifyRequests.
                         * Loop until Ctrl-C.
                         */
                        notifysignal = 1L << notifyport->mp_SigBit;
                        for (;;)
                        {
                            /* Wait for message port signals and break */
                            signal = Wait(notifysignal | SIGBREAKF_CTRL_C);

                            if (signal & notifysignal)
                            {
                                while (notifymsg =
                                        (struct NotifyMessage *) GetMsg(notifyport))
                                {
                                    /*
                                     * Here is that node again, stuffed in
                                     * nr_UserData. Can immediately reference
                                     * its data, like comparing filesize with
                                     * stored filesize in node, or remove it
                                     * (after an EndNotify() ofcourse).
                                     */
                                    Printf(
                                        "Notification message for %s, Node at 0x%lx\n",
                                        (LONG) notifymsg->nm_NReq->nr_Name,
                                        (LONG) notifymsg->nm_NReq->nr_UserData);
                                    ReplyMsg((struct Message *) notifymsg);
                                }
                            }

                            if (signal & SIGBREAKF_CTRL_C)
                            {
                                /*
                                 * Walk down the list, remove all
                                 * NotifyRequests, free all nodes.
                                 */
                                nnode = (struct NotifyNode *) notifylist->lh_Head;
                                while (nextnode =
                                        (struct NotifyNode *) nnode->nn_Node.ln_Succ)
                                {
                                    notifyrequest = &(nnode->nn_NotifyRequest);
                                    Printf("Removing notifcation for %s\n",
                                           (LONG) notifyrequest->nr_Name);
                                    /*
                                     * remove this request. EndNotify() will
                                     * also remove any messages on the message
                                     * port.
                                     */
                                    EndNotify(notifyrequest);
                                    /*
                                     * Not really needed to Remove() the node,
                                     * never going to use this list again
                                     */
                                    Remove((struct Node *) nnode);
                                    FreeMem(nnode, sizeof(struct NotifyNode));
                                    nnode = nextnode;
                                }
                                break;
                            }
                        }
                    }
                    FreeMem(notifylist, sizeof(struct List));
                }
                DeleteMsgPort(notifyport);
            }
            FreeArgs(readargs);
        }
        else
            PrintFault(IoErr(), NULL);
        CloseLibrary(DOSBase);
    }
}
```