

```

/* Part.c - AmigaMail File/Path separator example. Compiled with SAS/C 5.10a
lc -cfis -v -d0 -b0 -j73 Part.c
blink from Part.o to Part lib lib:amiga.lib ; if you don't have pragmas
quit
*/
#include <exec/types.h>
#include <exec/memory.h>
#include <dos/dos.h>
#include <dos/dosextens.h>
#include <dos/rdextens.h>

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

/* def PRAGMAS if you have them */
/* #define PRAGMAS */
#ifdef PRAGMAS
#include <pragmas/exec_pragmas.h>
#include <pragmas/dos_pragmas.h>
#else
struct ExecBase *SysBase;
struct Library *DOSBase;

#endif

VOID main(VOID);
LONG GetPath(UBYTE * path, UBYTE * buffer, LONG buffersize);
UBYTE *ItsWild(UBYTE * string);

VOID main(VOID)
{
#ifdef PRAGMAS
    struct Library *DOSBase;
#endif

    struct RDArgs *readargs;
    LONG rargs[2];
    LONG vargs[8];
    UBYTE *path, *filename;
    UBYTE *buffer;
    UBYTE *filepart, *pathpart;
    struct Process *process;
    BPTR lock;
    APTR wptr;
    BOOL error;

#ifdef PRAGMAS
    /* set up SysBase */
    SysBase = (*(struct Library **) 4));
#endif

    /* Fail silently if < 37 */
    if (DOSBase = OpenLibrary("dos.library", 37))
    {
        /*
         * Use a generous 256 byte buffer. Should suffice for everything but
         * extreme cases.
         */
        if (buffer = AllocMem(256, MEMF_CLEAR))
        {
            if (readargs = ReadArgs("PATH/A,FILENAME/A", rargs, NULL))
            {
                path = (UBYTE *) (rargs[0]);
                filename = (UBYTE *) (rargs[1]);

                error = GetPath(path, buffer, 255);
                if (error)
                    PrintFault(error, NULL);

                filepart = FilePart(path);
                pathpart = PathPart(path);

                vargs[0] = (LONG) path;
                vargs[1] = (LONG) filepart;
                vargs[2] = (LONG) pathpart;
                vargs[3] = (LONG) buffer;
            }
        }
    }
}

```

```

VPrintf(Output(),
        "Filename: %s\nFilepart: %s\nPathpart: %s\nPath: %s\n"
        vars);

/* No requesters */
process = (struct Process *) FindTask(NULL);
wptr = process->pr_WindowPtr;
process->pr_WindowPtr = (APTR) - 1L;

/*
 * Make sure this name is for real. This will weed out names
 * like "dh0:" and non-existent directories. (and also
 * complain about non-mounted volumes.) It is tempting to look
 * for trailing slashes and remove them but you shouldn't. You
 * might misinterpret the users intention. Better to generate a
 * warning and prompt for new input.
 */
if (lock = Lock(buffer, SHARED_LOCK))
    UnLock(lock);
else
    PrintFault(IoErr(), buffer);

/* Reset windowpointer */
process->pr_WindowPtr = wptr;

/*
 * Normally we should respect the test for an invalid path. To
 * show the results however, we blunder along...
 *
 * Add the filename to the path.
 */
if (AddPart(buffer, filename, 255))
    vars[0] = (LONG) buffer;
else
    vars[0] = (LONG) "OVERFLOW";

VPrintf(Output(), "\nNew path: %s\n", vars);

FreeArgs(readargs);
}
else
    PrintFault(IoErr(), NULL);
FreeMem(buffer, 256);
}
CloseLibrary(DOSBase);
}
}

/*
 * Standalone function to isolate a path and copy it into a supplied buffer.
 * Does not test if the path is valid. Returns an error in case of buffer
 * overflow.
 */
LONG
GetPath(UBYTE * path, UBYTE * buffer, LONG buffersize)
{
    UBYTE      *pathpart, *filepart;
    UBYTE      *tmp1, *tmp2;
    BPTR       lock;
    struct FileInfoBlock *fib;
    LONG       error = 0;

    /* Open own copy of dos.library if pragmas are used so it's standalone */
#ifdef PRAGMAS
    struct Library *DOSBase;

    if (! (DOSBase = OpenLibrary("dos.library", 36)))
        return (1);
#endif

    /*
     * If there seems to be no path, the pathpart will point to the filepart
     * too, so we need to check for that.
     */
    filepart = FilePart(path);
    pathpart = PathPart(path);

    /*

```

```

/* This also handles cases where there is only a volume/device name, only a
 * directory name or a combo of those.
 */
if (pathpart == path)
{
    /*
     * There seems to be only one component. Copy it if it is not wild.
     * Caller will have to check whether if it exists and if it is a file
     * or directory.
     */
    if (!(ItsWild(pathpart)))
        pathpart = NULL;
}

if (pathpart != path)
{
    /*
     * If pathpart equals filepart (pointer wise) then there is only one
     * component (possible preceded by a volume name).
     */
    if (pathpart == filepart)
    {
        if (!(ItsWild(pathpart)))
            pathpart = NULL;
    }
    else
    {
        /*
         * Try to lock it to determine if the last component is a
         * directory.
         */
        if (lock = Lock(path, SHARED_LOCK))
        {
            if (fib = AllocMem(sizeof(struct FileInfoBlock), MEMF_CLEAR))
            {
                if ((Examine(lock, fib)) == DOSTRUE)
                {
                    /* Hey it's a directory after all */
                    if (fib->fib_DirEntryType > 0)
                        pathpart = NULL;
                }
                FreeMem(fib, sizeof(struct FileInfoBlock));
            }
            Unlock(lock);
            /* else treat it as a filename */
        }
    }

    /* Copy the pathpart in the buffer */
    tmp1 = buffer;
    tmp2 = path;
    while ((*tmp1++ = *tmp2++) && (tmp2 != pathpart))
    {
        if (tmp1 == (buffer + buffersize))
        {
            error = ERROR_NO_FREE_STORE;
            break;
        }
    }
    *tmp1 = '\0'; /* NULL terminate. */
}

#ifdef PRAGMAS
    CloseLibrary(DOSBase);
#endif
return (error);
}

/* Simple test whether a filename contains wildcards or not */
UBYTE
ItsWild(UBYTE * string)
{
    static UBYTE *special = ".*?*[|";
    UBYTE *tmp = string;
    COUNT i;

```

```

do
{
    for (i = 0; special[i] != '\0'; i++)
    {
        if (*tmp == special[i])
            return (tmp);
    }
    tmp++;
} while (*tmp);

return (NULL);
}

```

/* **Split.c** - AmigaMail SplitName() example. Compiled with SAS/C 5.10a.
 lc -cfis -v -d0 -b0 -j73 Split.c
 blink from Split.o to Split lib lib:amiga.lib ; if you don't have pragmas
 quit

```

* Tuesday, 16-Jul-91 12:15:49, Ewout
*
*
*/
#include <exec/memory.h>
#include <dos/dosextens.h>
#include <dos/rdargs.h>

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

/* def PRAGMAS if you have them */
/* #define PRAGMAS */
#ifdef PRAGMAS
#include <pragmas/exec_pragmas.h>
#include <pragmas/dos_pragmas.h>
#else
struct ExecBase *SysBase;
struct DosLibrary *DOSBase;

#endif

#define BUFFERSIZE      128

VOID      main(VOID);

VOID
main(VOID)
{
#ifdef PRAGMAS
    struct DosLibrary *DOSBase;
#endif

    struct RDArgs    *readargs;
    LONG              rargs[2];
    UBYTE             *filename, *buffer;
    ULONG             buffersize;
    WORD              position = 0;
    LONG              vargs[4];

#ifdef PRAGMAS
    /* set up SysBase */
    SysBase = *((struct Library **) 4));
#endif

    /* Fail silently if < 37 */
    if (DOSBase = (struct DosLibrary *) OpenLibrary("dos.library", 37))
    {
        /* See the DOS Autodocs for more information about ReadArgs() */
        if (readargs = ReadArgs("FILE/A,BUFFERSIZE/A/N", rargs, NULL))
        {
            filename = (UBYTE *) rargs[0];
            buffersize = *((LONG *) rargs[1]);
            if (buffersize < 1 || buffersize > 4096)
                buffersize = BUFFERSIZE;

            if (buffer = AllocMem(buffersize, MEMF_CLEAR))
            {
                position = SplitName(filename, ':', buffer, position, buffersize);

                vargs[0] = position;
                vargs[1] = (LONG) buffer;
                VFPrintf(Output(), "Devicename: position: %ld Buffer: %s\n", vargs);

                if (position == -1)
                    position = 0;

                do
                {
                    position =
                        SplitName(filename, '/', buffer, position, buffersize);
                    vargs[0] = position;
                    vargs[1] = (LONG) buffer;
                    VFPrintf(Output(),
                        "Path component: position: %ld Buffer: %s\n",
                        vargs);
                } while (position != -1);
            }
        }
    }
}

```

```

        FreeMem(buffer, buffersize);
    }
    FreeArgs(readargs);
}
else
    PrintFault(IoErr(), NULL);
CloseLibrary((struct Library *) DOSBase);
}
}

```

§