$SPAD/src/lib cfuns-c.c

The Axiom Team

December 3, 2016

Abstract

Contents

1 License

```
/*
Copyright (c) 1991-2002, The Numerical ALgorithms Group Ltd.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

    - Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.

    - Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in
      the documentation and/or other materials provided with the
      distribution.

    - Neither the name of The Numerical ALgorithms Group Ltd. nor the
      names of its contributors may be used to endorse or promote products
      derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/
```

—— * ——

```
#include <stdio.h>
```

————-

The MACOSX platform is broken because no matter what you do it seems to include files from [[/usr/include/sys]] ahead of [[/usr/include]]. On linux systems these files include themselves which causes an infinite regression of includes that fails. GCC gracefully steps over that problem but the build fails anyway. On MACOSX the [[/usr/include/sys]] versions of files are badly broken with respect to the [[/usr/include]] versions.

—— * ——

```
#if defined(MACOSXplatform)
#include "/usr/include/unistd.h"
#else
#include <unistd.h>
#endif
#include <stdlib.h>
#include <string.h>
#if !defined(BSDplatform)
#include <malloc.h>
#endif
#include <sys/types.h>
#include <sys/stat.h>

#include "cfuns-c.h1"
```

———-

The `addtopath` function is used in `interp/i-toplev.boot` as part of the `start` function.

—— * ——

```
int addtopath(char *dir) {
    char *path, *newpath;
    path = getenv("PATH");
    if (path == NULL)
        return -1;
    newpath = (char *)
       malloc(1 + strlen(path) + strlen(dir) + strlen("PATH=:"));
    if (newpath == NULL)
        return -1;
    sprintf(newpath, "PATH=%s:%s", path, dir);
    return putenv(newpath);
}
```

———-

Test whether the path is the name of a directory. Returns 1 if so, 0 if not, -1 if it doesn't exist.

—— * ——

```
int directoryp(char *path) {
    struct stat buf;
    int code = stat(path, &buf);
    return(code == -1 ? -1 : S_ISDIR(buf.st_mode));
}
```

———-

This function is only used internal to this file. Axiom lisp code does not depend on it.

— * —

```
int make_path_from_file(char *s, char *t) {
    char *pos = "";
    char *c;
    /** simply copies the path name from t into s **/
    for (c = t + strlen(t); c != s; c--)
        if (*c == '/') {
            pos = c;
            break;
        }
    /** Check to see if the path was actually present **/
    if (c == t) {               /** No Path, so return the pwd **/
        return (-1);
    }
    /** now just do the copying **/
    strncpy(s, t, pos - t);
    return 1;
}
```

———-

This function is used in `interp/fname.lisp` to support the `myWriteable?` function, which is called by `fnameWriteable?`. It supports a test called `writeable?` in `algebra/fname.spad`.

— * —

```
int writeablep(char *path) {
    struct stat buf;
    char newpath[100];
    int code;
    code = stat(path, &buf);
    if (code == -1) {
        /** The file does not exist, so check to see
                 if the directory is writable                 *****/
        if (make_path_from_file(newpath, path) == -1 ||
            stat(newpath, &buf) == -1) {
            return (-1);
        }
        else {
            if (geteuid() == buf.st_uid) {
                return (2 * ((buf.st_mode & S_IWUSR) != 0));
            }
            else if (getegid() == buf.st_gid) {
                return (2 * ((buf.st_mode & S_IWGRP) != 0));
            }
```

```
            else {
                return (2 * ((buf.st_mode & S_IWOTH) != 0));
            }
        }
    }
    else if (geteuid() == buf.st_uid) {
        return ((buf.st_mode & S_IWUSR) != 0);
    }
    else if (getegid() == buf.st_gid) {
        return ((buf.st_mode & S_IWGRP) != 0);
    }
    else {
        return ((buf.st_mode & S_IWOTH) != 0);
    }
}
```

———-

This function does not appear to be used anywhere

```
int CLgetpid(void) {
     return getpid();
}
```

This function does not appear to be used in axiom. It has been replaced by native lisp code in `fname.lisp` in the function `file-readablep`.

```
int readablep(char *path) {
    struct stat buf;
    int code;
    code = stat(path, &buf);
    if (code == -1) {
        return (-1);
    }
    else if (geteuid() == buf.st_uid) {
        return ((buf.st_mode & S_IREAD) != 0);
    }
    else if (getegid() == buf.st_gid) {
        return ((buf.st_mode & S_IRGRP) != 0);
    }
    else {
        return ((buf.st_mode & S_IROTH) != 0);
    }
}
```

This function does not appear to be used anywhere.

```
long findString(char *file, char *string) {
    int nstring, charpos;
    FILE *fn;
    char buffer[1024];
```

```
    if ((fn = fopen(file, "r")) == NULL)
        return -1;
    for (charpos = 0, nstring = strlen(string);
         fgets(buffer, sizeof buffer, fn) != NULL;
         charpos += strlen(buffer)
        )
        if (!strncmp(buffer, string, nstring))
            return charpos;
    return -1;
}
```

This function does not appear to be used anywhere.

```
int copyEnvValue(char *varName, char *buffer) {
    char *s;
    s = getenv(varName);
    if (s == NULL)
        return 0;
    strcpy(buffer, s);
    return strlen(s);
}
```

References

[1] nothing