

fnkdat

Name

fnkdat — get a directory name

Synopsis

```
#include "fnkdat.h"
int fnkdat ( const char* target , char* buffer , int len , int flags );
```

INTRODUCTION

system-independent, adj.:
Works equally poorly on all systems.
-- unknown

Fnkdat provides an interface for determining common directory names in Win32 and UNIX applications.

Applications organize files differently on different platforms. On Win32 systems, an application is generally installed by a user and it is usually (more-or-less) installed entirely under a single tree. For example, the game OmniHype might be installed in the directory `C:\Program Files\OmniHype\`. Its maps or data may be in `C:\Program Files\OmniHype\maps\`, and the system configuration might be `C:\Program Files\OmniHype\config.dat`. Information specific to the user straczynski (such as keyboard/mouse settings) might be stored under `G:\Documents and Settings\Straczynski\Application Data\OmniHype\` in Windows NT and its derivatives, and under `C:\Program Files\OmniHype\users\straczynski\` in Windows 95 and its derivatives.

On a UNIX system the story is completely different. An application may be compiled by a user or a packager, and file locations (at least one of them, anyway) are often defined at compile-time. The application is then installed by a user or a system administrator, relative to a prefix (such as `/usr/`). Following the above example, the binary for the game OmniHype may be installed as `/usr/bin/omnihype`. Its maps or data may be in `/usr/share/omnihype/maps/`. The system configuration may be `/etc/omnihype.conf`, and information specific to the user straczynski may be stored under `~straczynski/.omnihype/`.

fnkdat is an attempt to cope with these differences.

USAGE

`fnkdat` is implemented as a single function. The source is released under a GUILF style license and may be statically linked to both public and commercial software. The license text, links to the source, and a couple tutorials are listed below.

PLATFORMS

`fnkdat` has been tested, and runs on the following platforms:

- Microsoft Windows NT/95/2000
- Cygwin (DLL version 1.3.10)
- x86 GNU/Linux 2.2 (Debian 2.2)
- x86 GNU/Linux 2.4 (RedHat 6.2, RedHat 7.2)
- x86 FreeBSD 4.5-STABLE
- Alpha GNU/Linux 2.2 (Debian 2.2)
- PPC - RS/6000 Linux 2.2 (Debian 2.2)
- Sparc GNU/Linux 2.2 (Debian 2.2)

`fnkdat` compiles on the following, though I doubt very much that the values being returned are correct. Patches welcome.

- PPC - G4 MacOSX Darwin Kernel Version 5.3
- Sparc - R220 Sun Solaris (8), SPROcc

DESCRIPTION

`fnkdat` accepts a *buffer* along with the length of the buffer (*len*) and fills it with a path name (directory or file name), as determined by *flags*. *target* may be a path name or NULL. If it is an absolute path name, then `fnkdat` simply copies *target* into *buffer* and returns. Otherwise (if it is relative) it's appended onto the end of *buffer* before `fnkdat` returns.

`fnkdat` is controlled by the bitwise `or` of one or more *flags*.

`FNKDAT_INIT`

Initialize `fnkdat`.

`fnkdat` should be called with this flag, once, before any other call is made.

`FNKDAT_UNINIT`

Uninitialize `fnkdat`.

`fnkdat` should be called with this flag, once, before the application terminates.

`FNKDAT_CONF`

`fnkdat` will fill *buffer* with the application's system-wide configuration directory.

As stated in the section File Usage (http://www.gnu.org/prep/standards_21.html#SEC21) of the *GNU Coding Standards*

Programs should be prepared to operate when `/usr` and `/etc` are read-only file systems. In general, this directory (and its contents) should be used to access configuration data that was installed with the application and need be manipulated by a system administrator (such as daemon configuration). For static configuration data, or data that's authored by someone else (such as a game's dungeon layout), `FNKDAT_DATA` is probably more appropriate.

`FNKDAT_USER`

`fnkdat` will fill *buffer* with the current user's directory for the application.

This is best used for finding a directory that's specific to both the current application and the current user. It's assumed that the application will use this directory to create files at run-time, and that its only requirements are that the directory be consistent and writable.

`FNKDAT_DATA`

`fnkdat` will fill *buffer* with the application's data directory.

This should be used to access read-only data that was installed with the application. It's a good place to store media files, such as graphics and sounds.

`FNKDAT_VAR` | `FNKDAT_DATA`

`fnkdat` will fill *buffer* with the application's variable data directory.

This should be use to store files that are created at run-time, but non-user specific; a "high scores" file, for example.

An additional flag may be bitwise `ored` with the above: `FNKDAT_CREAT`. `FNKDAT_CREAT` may be used to tell `fnkdat` to create any necessary directories (and parent directories), including those specified in *target*. It's similar to running the command `mkdir -p`. If *target* ends with a `/` character, then it is assumed to be a directory (and will be created if necessary). Otherwise, the last `/` character will be sought, and the directory that it names (as well as all its parents) will be created if necessary.

Also note that with the exception of directories determined via `FNKDAT_USER`, applications should not assume that the paths returned will be unique as they may be the same under win32.

RETURN VALUE

On success, zero is returned. On error, -1 is returned and `errno` is set appropriately.

ERRORS

ENOMEM

buffer is too small, and can not hold the entire result

EINVAL

flags is invalid

WIN32 Specifics

In Windows 95 and Windows 98, run-time generated files, such as user specific keymaps and highscore information, could be stored relative to an application binary. The game Starcraft does this, for example. However, with the creation of the `SHGetSpecialFolderPath` and `SHGetFolderPath` functions. Microsoft seems to be providing the ability for applications to be installed in read-only locations, while providing a means of determining directories for storing run-time and/or dynamic data. Now this is all well-and-good, but unless an application writer compiles with the EULA and includes the Microsoft "redistributable" `SHFOLDER.DLL` with his/her application, an application can not use `SHGetFolderPath`.

`fnkdat` will try to bind the `SHGetFolderPath` function at run-time. If the bind succeeds, `fnkdat` will use `SHGetFolderPath` where appropriate. If the bind fails, `fnkdat` will assume that the application is running on an older Microsoft OS (like Windows 95) and fallback to using directories that are relative to the running binary.

EXAMPLES

```
/*
```

```
UNIX
    /etc/omnihype/

WIN32
    C:\Program Files\OmniHype\
*/
char buffer[FILENAME_MAX];
fnkdat(NULL, buffer, FILENAME_MAX, FNKDAT_CONF);

/*
UNIX
    /home/jms/.omnihype/

Win32+shfolder.dll
    C:\Documents and Settings\Jms\Application Data\OmniHype\

WIN32
    C:\Program Files\OmniHype\Users\Jms\
*/
char buffer[FILENAME_MAX];
fnkdat(NULL, buffer, FILENAME_MAX, FNKDAT_USER);

/*
UNIX
    /etc/omnihype/server.conf

WIN32
    C:\Program Files\OmniHype\server.conf
*/
char buffer[FILENAME_MAX];
fnkdat("server.conf", buffer, FILENAME_MAX, FNKDAT_CONF);

/*
UNIX
    /home/jms/.omnihype/keymap.xml

Win32+shfolder.dll
    C:\Documents and Settings\Jms\Application Data\OmniHype\keymap.xml

WIN32
    C:\Program Files\OmniHype\Users\Jms\keymap.xml
*/
char buffer[FILENAME_MAX];
fnkdat("keymap.xml", buffer, FILENAME_MAX, FNKDAT_USER);

/*
UNIX
    /usr/share/omnihype/

WIN32
    C:\Program Files\OmniHype\
```

```
*/
char buffer[FILENAME_MAX];
fnkdat(NULL, buffer, FILENAME_MAX, FNKDAT_DATA);

/*
  UNIX
    /usr/share/omnihype/logo.jpg

  WIN32
    C:\Program Files\OmniHype\logo.jpg
*/
char buffer[FILENAME_MAX];
fnkdat("logo.jpg", buffer, FILENAME_MAX, FNKDAT_DATA);

/*
  UNIX
    /var/lib/games/omnihype/highscore

  WIN32+shfolder.dll
    G:\Documents and Settings\All Users\Application Data\OmniHype\highscore

  WIN32
    C:\Program Files\OmniHype\Users\Default\highscore
*/
char buffer[FILENAME_MAX];
fnkdat("highscore", buffer, FILENAME_MAX, FNKDAT_VAR | FNKDAT_DATA);
```

DOWNLOAD

You can get `fnkdat` as a tarball, zip, or RedHat RPM. `fnkdat-0.0.8.tar.gz`, `fnkdat-0.0.8.zip`

MANUAL

The `fnkdat` manpage is available in several formats: HTML (`fnkdat.html`), PS (`fnkdat.ps`), PDF (`fnkdat.pdf`), TXT (`fnkdat.txt`)

TUTORIALS

fnkapp

HTML (fnkapp.html), PS (fnkapp.ps), PDF (fnkapp.pdf), TXT (fnkapp.txt) A tutorial of how to use `fnkdat` with a simple Makefile. The complete tutorial is included in the distributions listed above.

VERSION

fnkdat-0.0.8

WEBPAGE

This project is hosted by the GNU Savannah Website; Patches, mailing lists, and distributions can be found there. <http://savannah.nongnu.org/projects/fnkdat/>.

You can join the mailing lists here http://savannah.gnu.org/mail/?group_id=1650.

The latest version, additional information, tutorials, and source can be found on the project homepage at <http://www.maccormack.net/~djm/fnkdat/>

AUTHOR

Written by David MacCormack

REPORTING BUGS

Report bugs to djm at maccormack dot net

COPYING

fnkdat - an interface for determining common directory names

Copyright (C) 2001, 2002 David MacCormack

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but **WITHOUT ANY WARRANTY**; without even the implied warranty of **MERCHANTABILITY** or **FITNESS FOR A PARTICULAR PURPOSE**. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

As a special exception, David MacCormack gives permission for additional uses of the text contained in the files named “fnkdat.h” and “fnkdat.c”, hereafter known as FNKDAT.

The exception is that, if you link the FNKDAT with other files to produce an executable, this does not by itself cause the resulting executable to be covered by the GNU General Public License. Your use of that executable is in no way restricted on account of linking the FNKDAT code into it.

This exception does not however invalidate any other reasons why the executable file might be covered by the GNU General Public License.

This exception applies only to the code released by David MacCormack under the name FNKDAT. If you copy code from other software into a copy of FNKDAT, as the General Public License permits, the exception does not apply to the code that you add in this way. To avoid misleading anyone as to the status of such modified files, you must delete this exception notice from them.

If you write modifications of your own for FNKDAT, it is your choice whether to permit this exception to apply to your modifications. If you do not wish that, delete this exception notice.

David MacCormack (djm at maccormack dot net)

FILES

fnkdat.h, fnkdat.c, fnkdat.sgm, ChangeLog