

fnkapp

Name

fnkapp — A fnkdat tutorial

Synopsis

fnkapp

INTRODUCTION

This is a tutorial on how to include `fnkdat` in a project using a plain 'ol Makefile. It's certainly not the only way to do it, it's just one way. If, after reading this, you can build and run a simple application that uses `fnkdat`, then I was successful ;).

SOURCE

The first thing you need to do is make a directory to store the code for this masterpiece. Since the application will be called `fnkapp`, `mkdir fnkapp` would seem appropriate. Next `cd fnkapp` and grab `fnkdat.h` & `fnkdat.c`, and save them. Next create a new file named `fnkapp.c` using your favorite editor, and put the following source in it.

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

int main(int argc, char** argv) {

    char buffer[FILENAME_MAX];

    /* STEP #1 */
    if (fnkdat(NULL, NULL, 0, FNKDAT_INIT) < 0) {
        perror("could not initialize fnkdat");
        return 2;
    }

    /* STEP #2 */
    if (fnkdat(NULL, buffer, FILENAME_MAX, FNKDAT_USER) < 0) {
        perror("could not get directory : FNKDAT_USER");
        return 2;
    }
    printf("FNKDAT_USER: %s\n", buffer);

    /* STEP #3 */
    if (fnkdat(NULL, buffer, 0, FNKDAT_UNINIT) < 0) {
```

```

        perror("could not uninitialized fnkdat");
        return 2;
    }

    return 0;
}

```

I'm assuming you're comfortable w/ C, so I'm only going to explain the `fnkdat` specific steps.

1. The first thing you need to do is initialize `fnkdat`. This is done by calling `fnkdat` with the `FNKDAT_INIT` flag. All other arguments are simply ignored.
2. Next, we use the library. This will store the current user's directory (for this application) in `buffer`.
3. Next we shutdown `fnkdat`. You should call this once you're finished using the library.

MAKEFILE

Now it's time to create a `Makefile`; copy the following text into it.

```

CFLAGS=-DPACKAGE=\"fnkapp\"

all: fnkapp

check: fnkapp
    ./fnkapp

fnkapp: fnkdat.o fnkapp.o
    $(CC) -o $@ *.o

clean:
    /bin/rm -f *.o fnkapp

```

This is pretty standard, and the only thing to explain is the extra define in the `CFLAFS`. The `PACKAGE` macro has the same purpose as the one generated by `autoheader` (if you're using `autoheader` in your project then you can use its output and omit this step). It's used throughout `fnkdat` in generating directory names. It should uniquely identify your application.

RUN

That's it! Go back to the command line and type **make**. You should see something along the lines of

```
cc -DPACKAGE=\"fnkapp\" -c -o fnkapp.o fnkapp.c
cc -o fnkapp *.o
```

Now type **./fnkapp** and you should see your new application tell you where it would store your user specific settings, if it had any. I see the following:

```
FNKDAT_USER: /home/djm/.fnkapp/
```

WEBPAGE

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