CFFileDescriptor Reference

Core Foundation > File Management

2007-05-23

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

CFFileDescriptor Reference

Derived From:	CFType
Framework:	CoreFoundation/CoreFoundation.h
Declared in	CFFileDescriptor.h

Overview

The CFFileDescriptor provides an opaque type to monitor file descriptors for read and write activity via CFRunLoop.

You use CFFileDescriptor to monitor file descriptors for read and write activity via CFRunLoop using callbacks. Each call back is one-shot, and must be re-enabled if you want to get another one.

You can re-enable the callback in the callback function itself, but you must completely service the file descriptor before doing so. For example, if you create a CFFileDescriptor for a pipe and get a callback because there are bytes to be read, then if you don't read all of the bytes but nevertheless re-enable the CFFileDescriptor for read activity, you'll get called back again immediately.

You can monitor kqueue file descriptors for read activity to find out when an event the kqueue is filtering for has occurred. You are responsible for understanding the use of the kevent() API and inserting and removing filters from the kqueue file descriptor yourself.

The following example takes a UNIX process ID as argument, and watches up to 20 seconds, and reports if the process terminates in that time:

```
// cc test.c -framework CoreFoundation -O
#include <CoreFoundation/CoreFoundation.h>
#include <unistd.h>
#include <sys/event.h>
static void noteProcDeath(CFFileDescriptorRef fdref, CFOptionFlags callBackTypes,
 void *info) {
    struct kevent kev;
    int fd = CFFileDescriptorGetNativeDescriptor(fdref);
    kevent(fd, NULL, 0, &kev, 1, NULL);
    // take action on death of process here
    printf("process with pid '%u' died\n", (unsigned int)kev.ident);
    CFFileDescriptorInvalidate(fdref);
   CFRelease(fdref); // the CFFileDescriptorRef is no longer of any use in this
 example
}
// one argument, an integer pid to watch, required
int main(int argc, char *argv[]) {
    if (argc < 2) exit(1);
    int fd = kqueue();
    struct kevent kev;
```

```
    EV_SET(&kev, atoi(argv[1]), EVFILT_PROC, EV_ADD|EV_ENABLE, NOTE_EXIT, 0,
NULL);
    kevent(fd, &kev, 1, NULL, 0, NULL);
    CFFileDescriptorRef fdref = CFFileDescriptorCreate(kCFAllocatorDefault, fd,
 true, noteProcDeath, NULL);
    CFFileDescriptorEnableCallBacks(fdref, kCFFileDescriptorReadCallBack);
    CFRunLoopSourceRef source =
CFFileDescriptorCreateRunLoopSource(kCFAllocatorDefault, fdref, 0);
    CFRunLoopAddSource(CFRunLoopGetMain(), source, kCFRunLoopDefaultMode);
    CFRelease(source);
    // run the run loop for 20 seconds
    CFRunLoopRunInMode(kCFRunLoopDefaultMode, 20.0, false);
    return 0;
}
```

Functions by Task

Creating a CFFileDescriptor

`CFFileDescriptorCreate` (page 7)
Creates a new CFFileDescriptor.

Getting Information About a File Descriptor

`CFFileDescriptorGetNativeDescriptor` (page 10)
Returns the native file descriptor for a given CFFileDescriptor.

`CFFileDescriptorIsValid` (page 11)
Returns a Boolean value that indicates whether the native file descriptor for a given CFFileDescriptor is valid.

`CFFileDescriptorGetContext` (page 9)
Gets the context for a given CFFileDescriptor.

Invalidating a File Descriptor

`CFFileDescriptorInvalidate` (page 10)
Invalidates the native file descriptor for a given CFFileDescriptor.

Managing Callbacks

`CFFileDescriptorEnableCallBacks` (page 9)
Enables callbacks for a given CFFileDescriptor.

`CFFileDescriptorDisableCallBacks` (page 8)
Disables callbacks for a given CFFileDescriptor.

Creating a Run Loop Source

`CFFileDescriptorCreateRunLoopSource` (page 8)
Creates a new runloop source for a given CFFileDescriptor.

Getting the CFFileDescriptor Type ID

`CFFileDescriptorGetTypeID` (page 10)
Returns the type identifier for the CFFileDescriptor opaque type.

Functions

CFFileDescriptorCreate

Creates a new CFFileDescriptor.

```
CFFileDescriptorRef CFFileDescriptorCreate (
    CFAllocatorRef allocator,
    CFFileDescriptorNativeDescriptor fd,
    Boolean closeOnInvalidate,
    CFFileDescriptorCallBack callout,
    const CFFileDescriptorContext *context
);
```

Parameters

allocator
The allocator to use to allocate memory for the new bag and its storage for values. Pass `NULL` or `kCFAllocatorDefault` to use the current default allocator.

fd
The file descriptor for the new CFFileDescriptor.

closeOnInvalidate
`true` if the new CFFileDescriptor should close *fd* when it is invalidated, otherwise `false`.

callout
The CFFileDescriptorCallBack for the new CFFileDescriptor.

context
Contextual information for the new CFFileDescriptor.

Return Value
A new CFFileDescriptor or `NULL` if there was a problem creating the object. Ownership follows the Create Rule.

Availability
Available in Mac OS X v10.5 and later.

See Also
`CFFileDescriptorGetContext` (page 9)
`CFFileDescriptorInvalidate` (page 10)

Declared In
`CFFileDescriptor.h`

CFFileDescriptorCreateRunLoopSource

Creates a new runloop source for a given CFFileDescriptor.

```
CFRunLoopSourceRef CFFileDescriptorCreateRunLoopSource (
    CFAllocatorRef allocator,
    CFFileDescriptorRef f,
    CFIndex order
);
```

Parameters

allocator

The allocator to use to allocate memory for the new bag and its storage for values. Pass `NULL` or `kCFAllocatorDefault` to use the current default allocator.

f

A CFFileDescriptor.

order

The order for the new run loop (see `CFRunLoopSourceCreate`).

Return Value
A new runloop source for *f*, or `NULL` if there was a problem creating the object. Ownership follows the Create Rule.

Discussion
The context for the new runloop (see `CFRunLoopSourceCreate`) is the same as the context passed in when the CFFileDescriptor was created (see `CFFileDescriptorCreate` (page 7)).

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CFFileDescriptor.h`

CFFileDescriptorDisableCallBacks

Disables callbacks for a given CFFileDescriptor.

```
void CFFileDescriptorDisableCallBacks (
    CFFileDescriptorRef f,
    CFOptionFlags callBackTypes
);
```

Parameters

f

A CFFileDescriptor.

callBackTypes

A bitmask that specifies which callbacks to disable (see "Callback Identifiers" (page 13) for possible components).

Availability
Available in Mac OS X v10.5 and later.

See Also
`CFFileDescriptorEnableCallBacks` (page 9)

Declared In
`CFFileDescriptor.h`

CFFileDescriptorEnableCallBacks

Enables callbacks for a given CFFileDescriptor.

```
void CFFileDescriptorEnableCallBacks (
    CFFileDescriptorRef f,
    CFOptionFlags callBackTypes
);
```

Parameters

f

A CFFileDescriptor.

callBackTypes

A bitmask that specifies which callbacks to enable (see "Callback Identifiers" (page 13) for possible components).

Availability
Available in Mac OS X v10.5 and later.

See Also
`CFFileDescriptorDisableCallBacks` (page 8)

Declared In
`CFFileDescriptor.h`

CFFileDescriptorGetContext

Gets the context for a given CFFileDescriptor.

```
void CFFileDescriptorGetContext (
    CFFileDescriptorRef f,
    CFFileDescriptorContext *context
);
```

Parameters

f

A CFFileDescriptor.

context

Upon return, contains the context passed to *f* in `CFFileDescriptorCreate` (page 7).

Availability
Available in Mac OS X v10.5 and later.

See Also
CFFileDescriptorCreate (page 7)

Declared In
CFFileDescriptor.h

CFFileDescriptorGetNativeDescriptor

Returns the native file descriptor for a given CFFileDescriptor.

```
CFFileDescriptorNativeDescriptor CFFileDescriptorGetNativeDescriptor (
    CFFileDescriptorRef f
);
```

Parameters

f

A CFFileDescriptor.

Return Value
The native file descriptor for *f*.

Availability
Available in Mac OS X v10.5 and later.

See Also
CFFileDescriptorInvalidate (page 10)

Declared In
CFFileDescriptor.h

CFFileDescriptorGetTypeID

Returns the type identifier for the CFFileDescriptor opaque type.

```
CFTypeID CFFileDescriptorGetTypeID (
    void
);
```

Return Value
The type identifier for the CFFileDescriptor opaque type.

Availability
Available in Mac OS X v10.5 and later.

Declared In
CFFileDescriptor.h

CFFileDescriptorInvalidate

Invalidates the native file descriptor for a given CFFileDescriptor.

```
void CFFileDescriptorInvalidate (
    CFFileDescriptorRef f,
);
```

Parameters

f

A CFFileDescriptor.

Availability
Available in Mac OS X v10.5 and later.

See Also
`CFFileDescriptorIsValid` (page 11)
`CFFileDescriptorGetNativeDescriptor` (page 10)

Declared In
`CFFileDescriptor.h`

CFFileDescriptorIsValid

Returns a Boolean value that indicates whether the native file descriptor for a given CFFileDescriptor is valid.

```
Boolean CFFileDescriptorIsValid (
    CFFileDescriptorRef f,
);
```

Parameters

f

A CFFileDescriptor.

Return Value
`true` if the native file descriptor for *f* is valid, otherwise `false`.

Availability
Available in Mac OS X v10.5 and later.

See Also
`CFFileDescriptorInvalidate` (page 10)

Declared In
`CFFileDescriptor.h`

Data Types

CFFileDescriptorNativeDescriptor

Defines a type for the native file descriptor.

```
typedef int CFFileDescriptorNativeDescriptor;
```

Availability
Available in Mac OS X v10.5 and later.

Declared In
CFFileDescriptor.h

CFFileDescriptorCallBack

Defines a structure for a callback for a CFFileDescriptor.

```
typedef void (*CFFileDescriptorCallBack) (
    CFFileDescriptorRef f,
    CFOptionFlags callBackTypes,
    void *info
);
```

Declared In
CFFileDescriptor.h

CFFileDescriptorContext

Defines a structure for the context of a CFFileDescriptor.

```
typedef struct {
    CFIndex    version;
    void *    info;
    void *    (*retain)(void *info);
    void    (*release)(void *info);
    CFStringRef    (*copyDescription)(void *info);
} CFFileDescriptorContext;
```

Fields

`version`
: The version number of this structure. If not one of the defined version numbers for this opaque type, the behavior is undefined. The current version of this structure is 0.

`info`

`retain`
: The retain callback used by the CFFileDescriptor.

`release`
: The release callback used by the CFFileDescriptor.

`copyDescription`
: The callback used to create a descriptive string representation of the CFFileDescriptor.

Availability
Available in Mac OS X v10.5 and later.

Declared In
CFFileDescriptor.h

CFFileDescriptorRef

A reference to an CFFileDescriptor object.

```
typedef struct __CFFileDescriptor * CFFileDescriptorRef;
```

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CFFileDescriptor.h`

Constants

Callback Identifiers

Constants that identify the read and write callbacks.

```
enum {
    kCFFileDescriptorReadCallBack = 1 << 0,
    kCFFileDescriptorWriteCallBack = 1 << 1
};
```

Constants

`kCFFileDescriptorReadCallBack`
Identifies the read callback.
Available in Mac OS X v10.5 and later.
Declared in `CFFileDescriptor.h`.

`kCFFileDescriptorWriteCallBack`
Identifies the write callback.
Available in Mac OS X v10.5 and later.
Declared in `CFFileDescriptor.h`.

Declared In
`CFFileDescriptor.h`

Document Revision History

This table describes the changes to *CFFileDescriptor Reference*.

Date	Notes
2007-05-23	New document that describes the opaque type to monitor file descriptors for read and write activity via CFRunLoop.

Document Revision History

Index

C

K