LogFileManagement

Compress your log files, or upload them to a server.

Introduction

When you think about it, there are two components of file logging. One component is the part that actually writes the log messages to the file, and decides when the file needs to be rolled. And the other component is the part that actually manages the log files. This is the part that actually does something with the log files once they're completed. Maybe it compresses them to save disk space. Maybe it uploads them to a server. Maybe both!

The DDFileLogger implementation, as you may now have guessed, is split into two components. DDFileLogger is the component that writes the log messages to the file. And DDLogFileManager is a protocol for managing log files, and deciding what to do with them after they've been rolled.

There are two ways to initialize a DDFileLogger instance:

```
@interface DDFileLogger : NSObject <DDLogger>
...

- (id)init;
- (id)initWithLogFileManager:(id <DDLogFileManager>)logFileManager;

...
@end
```

The default init method simply uses an instance of DDLogFileManagerDefault, which is a class that simply deletes old log files. (It keeps a configurable number around, but any past that amount are simply deleted.)

The alternative init method allows you to pass an instance of your own custom log file manager.

Log File Manager

Let's take a look at the DDLogFileManager protocol:

```
@protocol DDLogFileManager <NSObject>
@required

// Public properties

@property (readwrite, assign) NSUInteger maximumNumberOfLogFiles;

// Public methods

- (NSString *)logsDirectory;

- (NSArray *)unsortedLogFilePaths;
- (NSArray *)unsortedLogFileNames;
- (NSArray *)unsortedLogFileInfos;

- (NSArray *)sortedLogFilePaths;
- (NSArray *)sortedLogFileNames;
- (NSArray *)sortedLogFileInfos;

// Private methods (only to be used by DDFileLogger)

- (NSString *)createNewLogFile;

@optional

// Notifications from DDFileLogger

- (void)didArchiveLogFile:(NSString *)logFilePath
- (void)didArchiveLogFile:(NSString *)logFilePath;
- (void)didRollAndArchiveLogFile:(NSString *)logFilePath;

@end
```

There are methods to get the logs directory, and various methods to get the list of log files. Then there is a method to create a new log file (that returns the new log file path). And lastly, there are hooks from DDFileLogger to be notified of when a log file is rolled.

The hooks are designed to allow you do something with those archived log files!

The framework comes with a sample Xcode project that demonstrates compressing archived log files to save disk space. (It performs the compression using gzip.) The Xcode project name is "LogFileCompressor". The "CompressingLogFileManager" class is ready to be used in your project, or you can use it as a template, or simply learn from it so you can do your own custom log file management.