File Tracker Questions

A class or a module?

Because this problem involves calling methods and remembering the state, it would be a class problem.

Error handling? Robustness? Security? Are any of these required?

Error Handling:

We will treat any errors not related to files as extraordinary cases, log them, and continue execution. Errors will not cause our program to exit.

Robustness

We will sanitize inputs to prevent the user from tampering with the underlying system.

Security

We will prevent the user from watching files they do not have permission to access.

What components of the Ruby exception hierarchy are applicable to this problem? Illustrate your answer.

Exceptions that are applicable to this problem are:

- RuntimeError
- SecurityError
- ThreadError
- SystemCallError
- SystemStackError

These are all exceptions and errors that can be thrown when dealing with threading and system programming.

Does this problem require an iterator?

As we plan to delegate the problem solving to C code, we should not need to use an iterator.

Q: Describe Java's anonymous inner classes.

A: An anonymous inner class is a class that does not have a name and has been declared and instantiated simultaneously, inside of another class. They are used in order to

perform some specific action once. A common use case is adding an anonymous actionListener to GUI buttons.

Q: Compare and Contrast Java's anonymous inner classes and Ruby Proc objects; which do you think is better?

A: Both of these are methods of implementing a "closure". TODO more

From a cohesion viewpoint, which interface protocol is superior? Explain your decision!

From a cohesion standpoint, Ruby's Proc objects protocol is superior.
This is because Proc classes are more likely to be resuable, which leads to higher cohesion.

Is Module Errno useful in this problem? Illustrate your answer.

Module Errno would not be useful. Because we are delegating most of the system programming to the C code, having the Ruby code handle the system errors would be unnecessary.

Do any of the Anti-patterns described at:
http://today.java.net/pub/a/today/2006/04/06/exception-handling-antipatterns.html
Exist in your solution.

None of the anti-patterns described above exist in our solution.

Q: Describe the content of the library at:
http://c2.com/cgi/wiki?ExceptionPatterns

A: The library covers exception patterns. It covers patterns for defining exception types, raising exceptions, and handling exceptions. It also covers when you should or should not use exceptions and what some alternatives are.

Which are applicable to this problem? Illustrate your answer.
Which are applicable to the previous two problems? Illustrate your answer.

For the shell program, the Exception Reporter, Avoid Exceptions Whenever Possible, Code Without Exceptions patterns apply.
The shell program should handle all exceptions gracefully, returning errors to the user without stopping the shell.

Q: Is a directory, a file? Is a pipe, a file? Is a ....., a file? Tell us your thoughts on the definition of a file in a LINUX context.

A: A directory is a file that lists other files. A named pipe is special type of file. In the LINUX context almost everything on the system is a file. If it visible on the system it is a file.

Q: Define what is meant (in a LINUX environment) by file change? Does it mean only contents? Or does it include meta-information? What is meta-information for a file?

A: A file change triggers the last update time of a file to change. It only includes a change to the file's content. The meta information includes file ownership, permissions, last update time, last access time, number of hard links, size.