Domain Knowledge

- Shell

Design Questions

Module 1:

Reference

Our problem is a module. Modules do not need to be instantiated while classes need to be. We are also choosing modules because Modules are about giving us functions, which can be used for the shell and file watcher, while classes are about producing objects, which is something that is not required of us for this project.

What shell(s) are you using to provide a specification? What features do they support?

We are using the C Shell to provide a specification. Some features include history and editing, aliases, directory stacks, tilde notation, cdpath, job control and path hashing. This shell is chosen because of its primary features with history, cdpath and path hashing. The rest of the functions might be concidered but are not required.

In your opinion, which features are essential (should include in your design) and which are “window dressing” (should not include in your design)?

We believe that directory functions like ls and cd will be required functions for navigating and selecting files to monitor using the File Watcher. Tilde notation, aliases , history and editing are not needed for the completion of file watcher and will be considered as extra dressing that we might add if time permits.

Economics: Which, if any, essential features will be omitted from your design due to unmanageable effort requirements?

The directory stack, job control and path hashing will be omitted from the shell. Despite how useful they will be for a file watcher, they are not essential for its completion as functions like job control can be taken care of by the parent process in c.

Error handling? What percentage of code handles functional against potential pitfalls:

In the average commercial program? In your shell program?: 20% of code in the average commercial program handles error handling and about 10% of our code will handle error handling.

If they are radically different, please provide a rationale: Our shell will be performing most of the error handling in our program since it wil be handling the inputs. If we manage to sanitize and ensure that the \

Robustness? How do we make the system bullet-proof? Is Avoiding Core dumps of system shells important? Especially from a Security viewpoint, remember this dump will give access to underlying C system code and potentially Linux daemons?

We can make the system robust by validating the input commands and making sure that the user correctly inputs valid commands. We can stop core dumps by ensuring that there are correct inputs and files, and that links to the files have been closed.

Describe the Ruby exception hierarchy, which classes of exceptions are applicable to this problem?

Reference Ruby's exception heirarchy is as follows:

```
1) NoMemoryError
2) ScriptError
3) SignalException
```

```
4) StandardError
5) SystemExit
6) fatal
```

Standard error will be applicatble in order to handle IO exceptions. NoMemoryError will be needed due to the theoreticaly low amount of memory that our system will be having, especially if we start to create several children for each process instead of reusing our unused children. SignalException will be used to ensure that interupts from the child processes are handled correctly. SystemExit will be taken into consideration in order to clean up processes if the shell is closed unexpectedly.

What is Module Errno? Is it applicable to the problem? Explain your answer! Remember Ruby often wraps C code.

Errno Documentation

The OS reports errors using pain integers. Module Errno is created dynamically to map operating system errors to Ruby classe, with each error number generating its own subclass of SystemCallError. This module is applicable to our project, since there needs to be a way to trace errors generated from C code and bring it to the ruby's front end to be displayed to the user.

Security? How will we protect the system from tainted objects? Can we trust the user?

We can use ruby to detect and sanitize tainted objects to prevent the user from attacking the host system. We can never trust the user.

Is sand boxing applicable to this problem? Is it feasible to write security contracts?

Sandboxing should not be needed to run untrusted code because it has either been used thoroughly or we will write it ourselves. It is feasable to write security contracts to test for tainted inputs.

Should we be using class GetoptLong? Or Regexp? Or shell? Or ...

GetoptLong: parse conmmand line option. It is a pure ruby implementation.

Regexp: Used to match patterns against a string.

Shell: Implements an idiomatic interface for common UNIX shell commands.

Use Regexp since some of the command line options will be custom made.

What environment does a shell run within? Current Directory? Or ...

The shell will run within the current directory so the user can use cd to start navigation from the current directory.

What features should be user controllable? Prompts? Input and Output channels? Or ...

Inputs into the shell will be widely used, especially with file watcher. An output channel can be used if the user would want the data sent to a logfile instead of the shell.

Module 2:

A class or a module?

Likely, the implementation of this design would be completed through a series of modules, each holding the specific functions required. More specifically, the 2nd module would be housed within a module, and would be called by the main script by the child process.

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

The inputs would be sanitised beforehand by the Ruby script before being processed. This removes the possibility of 'tarnished' inputs prior to processing. Once the system is in module 2's exclusive code, the system follows the following parameters:

Error Handling: the system would protect against external interrupts/kill signals, which would terminate the program, as well as the child. Additionally, if the second module encounters an issue on its own (for example, if the system encounters a segmentation fault before the process is complete) the module must appropriately handle the situation.

Robustness: the system has to be robust enough to work despite adverse conditions. This means that the program would continue to work despite a segmentation fault raised outside of the process.

Security: the system needs to be isolated, strictly within the child process. As a result, the system can protect the child process from external control, and thus would result in a secure system.

These aspects are required, especially for a command line interface application. As users expect a level of quality with the innermost processes (aspects obtained by having a secure, robust system), developers need to build applications with respect to these key qualities at all times.

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

Reference:

- http://blog.nicksieger.com/articles/2006/09/06/rubys-exception-hierarchy/
- http://ruby-doc.org/core-2.1.1/Exception.html

The following components are applicable to this problem:

1. http://ruby-doc.org/core-2.1.1/ArgumentError.html
   - ArgumentErrors occur if an incorrect argument is passed. We can use to clean inputs.
2. http://ruby-doc.org/core-2.1.1/RuntimeError.html
   - Generic class used if the system encounters a runtime error.
3. http://ruby-doc.org/core-2.1.1/TypeError.html
   - TypeError when the system is passed a bad input (eg. RelayMessage "Two" "Hello World", where proper input is RelayMessage 2 "Hello World")

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

Reference:

- http://ruby-doc.org/core-2.1.1/Errno.html
- http://blog.honeybadger.io/understanding-rubys-strange-errno-exceptions/

Errno could potentially be useful to identify system errors - but more so for other modules. Important errno to take account of are:

1. EACCES Permission denied; the file permissions do not allow the attempted operation.
   - Didn't have permission to call the function.
2. EINVAL Invalid argument. This is used to indicate various kinds of problems with passing the wrong argument to a library function.
   - Invalid argument passed; likely caught by ruby function before processing.

Describe the article at:

http://today.java.net/pub/a/today/2006/04/06/exception-handlingantipatterns.html. Reference:

- https://community.oracle.com/docs/DOC-983543https://community.oracle.com/docs/DOC-983543

Used to identify exception handling, as well as custom exceptions. More specifically, the article focuses on antipatterns, which discuss 'bad' patterns for exceptions to have.

Convince the marker that these Anti-patterns don't exist in your solution.

We do not have any antipatterns in our solution. This is because of three reasons:

1. We do not plan on throwing exceptions in our C++ code. Instead, we plan on handling all of our interactions on the Ruby level, then passing only safe operations inside of the C code.
2. We have sleep operations, but they are so small that they would not ignore interrupts.
3. We do not return null on our functions.

Do they exist in your Shell solution?

No, our contracts control all of our sanitation - allowing us to be free of antipatterns. Additionally, the previous answer has been integrated into our design philosophy, which would ensure that no antipatterns exists.

How can I make the timing accurate? What time resolution should I be looking at, remember real-time systems? Time formats?

Generally, system time is considered accurate enough. For example, sleep(x) is considered accurate.

Does 'C' have better facilities for this problem than Ruby? (Big hint!)

Yes, C has better facilities for this problem. We plan on using one of those key facilities (sleep) to accurately time our processes.

What should be user controllable? Can we trust the user?

The values that should be user controllable are: evoking the function, setting the message, and setting the time duration. Because we properly sanitize the input, we can trust the user with manipulating the message, as well as setting the time duration. Of course, various gates are used in order to ensure that the user cannot 'break' the system - this includes adding a limit to both message size and duration, as well as preventing any 'bad' input from going through. However, because this module is naturally unintrusive, most functionality can be trusted to the user.

Module 3:

Which interface protocol is presented?

A class or a module?

This should be a module (sub-module). We do not need to have multiple instantiations. Any state data will be maintained by the encompassing module or class that is the shell.

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

The inputs would be sanitised beforehand by the Ruby script before being processed. This removes the possibility of 'tarnished' inputs prior to processing. Once the system is in module 2's exclusive code, the system follows the following parameters:

Error Handling: the system would protect against external interrupts/kill signals, which would terminate the program, as well as the child. Additionally, if the second module encounters an issue on its own (for example, if the system encounters a segmentation fault before the process is complete) the module must appropriately handle the situation.

Robustness: the system has to be robust enough to work despite adverse conditions. This means that the program would continue to work despite a segmentation fault raised outside of the process.

Security: the system needs to be isolated, strictly within the child process. As a result, the system can protect the child process from external control, and thus would result in a secure system.

These aspects are required, especially for a command line interface application. As users expect a level of quality with the innermost processes (aspects obtained by having a secure, robust system), developers need to build applications with respect to these key qualities at all times.

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

The ruby exception heirarchy can be explained here:

ruby Exception IRB::Abort MonitorMixin::ConditionVariable::Timeout NoMemoryError ScriptError LoadError Gem::LoadError NotImplementedError SyntaxError SecurityError SignalException Interrupt StandardError ArgumentError Gem::Requirement::BadRequirementError EncodingError Encoding::CompatibilityError Encoding::ConverterNotFoundError Encoding::InvalidByteSequenceError Encoding::UndefinedConversionError Exception2MessageMapper::ErrNotRegisteredException FiberError IOError EOFError IRB::CantChangeBinding IRB::CantReturnToNormalMode IRB::CantShiftToMultiIrbMode IRB::IllegalParameter IRB::IllegalRCGenerator IRB::IrbAlreadyDead IRB::IrbSwitchedToCurrentThread IRB::NoSuchJob IRB::NotImplementedError IRB::Notifier::ErrUndefinedNotifier IRB::Notifier::ErrUnrecognizedLevel IRB::OutputMethod::NotImplementedError IRB::SLex::ErrNodeAlreadyExists IRB::SLex::ErrNodeNothing IRB::UndefinedPromptMode IRB::UnrecognizedSwitch IndexError KeyError StopIteration LocalJumpError Math::DomainError NameError NoMethodError RangeError FloatDomainError RegexpError RubyLex::AlreadyDefinedToken RubyLex::SyntaxError RubyLex::TerminateLineInput RubyLex::TkReading2TokenDuplicateError RubyLex::TkReading2TokenNoKey RubyLex::TkSymbol2TokenNoKey RuntimeError Gem::Exception Gem::CommandLineError Gem::DependencyError Gem::DependencyRemovalException Gem::DocumentError

Gem::EndOfYAMLException Gem::FilePermissionError Gem::FormatException Gem::GemNotFoundException Gem::SpecificGemNotFoundException Gem::GemNotInHomeException Gem::InstallError Gem::InvalidSpecificationException Gem::OperationNotSupportedError Gem::RemoteError Gem::RemoteInstallationCancelled Gem::RemoteInstallationSkipped Gem::RemoteSourceException Gem::VerificationError SystemCallError ThreadError TypeError ZeroDivisionError SystemExit Gem::SystemExitException SystemStackError fatal

Does this problem require an iterator?

Yes, this problem requires an iterator. We have to use one to continually cycle through our operations, in order to ensure that the system is constantly 'examined' for FileWatcher.

Describe Java's anonymous inner classes.

Single succinct expression with no name often used if only requiring one instance of the class. Can be included in method calls.

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

Ruby Proc:

- Bound to local vars.

Java Anon Inner Class:

- They have a larger overhead by passsing the whole class

Ruby's proc is better because it operate within the class constraints without having to pass the whole class.

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

```
FileWatch( type of alteration, duration, list of filenames) {action}
```

Or

```
FileWatchCreation(duration, list of filenames) { action}
FileWatchAlter(duration, list of filenames) { action}
FileWatchDestroy(duration, list of filenames) { action}
```

Low Coupling and High Cohesion are the the attributes we find in well-structured, maintainable, high readable software. In general the more variables a method manipulates the more cohesive that method is to its class. Thus from a high cohesion viewpoint the superior interface protocol would be:

```
FileWatch( type of alteration, duration, list of filenames) {action}
```

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

Reference:

- http://ruby-doc.org/core-2.1.1/Errno.html

Operating systems report errors in an integer format. Using Errno we can identify and handle each System Error with a ruby class. This will be very useful in our problem when attempting to design a reliable and secure program.

1. EACCES Permission denied; the file permissions do not allow the attempted operation.
    - Didn't have permission to call the function.
2. EINVAL Invalid argument. This is used to indicate various kinds of problems with passing the wrong argument to a library function.
    - Invalid argument passed; likely caught by ruby function before processing.

ruby Errno.constants => [:NOERROR, :E2BIG, :EACCES, :EADDRINUSE, :EADDRNOTAVAIL, :EADV, :EAFNOSUPPORT, :EAGAIN, :EALREADY, :EAUTH, :EBADARCH, :EBADE, :EBADEXEC, :EBADF, :EBADFD, :EBADMACHO, :EBADMSG, :EBADR, :EBADRPC,

:EBADRQC, :EBADSLT, :EBFONT, :EBUSY, :ECANCELED, :ECHILD, :ECHRNG, :ECOMM, :ECONNABORTED, :ECONNREFUSED, :ECONNRESET, :EDEADLK, :EDEADLOCK, :EDESTADDRREQ, :EDEVERR, :EDOM, :EDOOFUS, :EDOTDOT, :EDQUOT, :EEXIST, :EFAULT, :EFBIG, :EFTYPE, :EHOSTDOWN, :EHOSTUNREACH, :EIDRM, :EILSEQ, :EINPROGRESS, :EINTR, :EINVAL, :EIO, :EIPSEC, :EISCONN, :EISDIR, :EISNAM, :EKEYEXPIRED, :EKEYREJECTED, :EKEYREVOKED, :EL2HLT, :EL2NSYNC, :EL3HLT, :EL3RST, :ELIBACC, :ELIBBAD, :ELIBEXEC, :ELIBMAX, :ELIBSCN, :ELNRNG, :ELOOP, :EMEDIUMTYPE, :EMFILE, :EMLINK, :EMSGSIZE, :EMULTIHOP, :ENAMETOOLONG, :ENAVAIL, :ENEEDAUTH, :ENETDOWN, :ENETRESET, :ENETUNREACH, :ENFILE, :ENOANO, :ENOATTR, :ENOBUFS, :ENOCSI, :ENODATA, :ENODEV, :ENOENT, :ENOEXEC, :ENOKEY, :ENOLCK, :ENOLINK, :ENOMEDIUM, :ENOMEM, :ENOMSG, :ENONET, :ENOPKG, :ENOPOLICY, :ENOPROTOOPT, :ENOSPC, :ENOSR, :ENOSTR, :ENOSYS, :ENOTBLK, :ENOTCONN, :ENOTDIR, :ENOTEMPTY, :ENOTNAM, :ENOTRECOVERABLE, :ENOTSOCK, :ENOTSUP, :ENOTTY, :ENOTUNIQ, :ENXIO, :EOPNOTSUPP, :EOVERFLOW, :EOWNERDEAD, :EPERM, :EPFNOSUPPORT, :EPIPE, :EPROCLIM, :EPROCUNAVAIL, :EPROGMISMATCH, :EPROGUNAVAIL, :EPROTO, :EPROTONOSUPPORT, :EPROTOTYPE, :EPWROFF, :EQFULL, :ERANGE, :EREMCHG, :EREMOTE, :EREMOTEIO, :ERESTART, :ERFKILL, :EROFS, :ERPCMISMATCH, :ESHLIBVERS, :ESHUTDOWN, :ESOCKTNOSUPPORT, :ESPIPE, :ESRCH, :ESRMNT, :ESTALE, :ESTRPIPE, :ETIME, :ETIMEDOUT, :ETOOMANYREFS, :ETXTBSY, :EUCLEAN, :EUNATCH, :EUSERS, :EWOULDBLOCK, :EXDEV, :EXFULL]

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

See Module 2 for response.

Reference:

- Effective Java Exceptions
- Exception-Handling Anti-patterns Blog
  - Log and Throw
  - Throwing Exception
  - Throwing the Kitchen Sink
  - Catching Exception
  - Destructive Wrapping
  - Log and Return Null
  - Catch and Ignore
  - Throw from Within Finally
  - Multi-Line Log Messages
  - Unsupported Operation Returning Null
  - IgnoringInterruptedException
  - Relying on getCause()

See Module 2 for response.

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

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

Current Problem

We are attempting to insert as few exceptions as possible in our code. Instead, we woul rather prefer to direct our users to correct paths, rather than breaking the system whenever a wrong path is accessed. This means that the following exception patterns are useful for us:

- http://c2.com/cgi/wiki?AvoidExceptionsWheneverPossible

We use this methodology to avoid exceptions, instead to provide references to issues and return false. This system is implemented both in our contracts and in our C structure.

- http://c2.com/cgi/wiki?LookBeforeYouLeap

We handle potential exceptions before they occur. This allows you to handle errors before they occur, which ensures user interaction for the user.

- http://c2.com/cgi/wiki?UseAssertions

We're using contracts in lieu of assertions, but the overall structure is the same. By doing so, we can ensure the safety of both pre- and post- conditions for our shell.

Previous Two Problems The same exception patterns are applicable for the previous two problems as well.

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.

Formally a file consists of an inode (file properties, incl pointer to data) and its data storage. In special storage is located the file names and directories.

Directory is a file. Pipe is not a file - it is a process.

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?

Linux Metadata

A file change occurs whenever any file metadata changes. It does not only mean contents, it also means the metadata, since a person can access the file which will change some information in the metadata and alter the file. For us, we are only conserned with the last modified time, since this will tell us when the file's contents have been altered.