project_specifications.md - Grip

Shells

Ruby Shell

Commands

Generic functions for shells:

```
mkdir (directory_name)               :: make the directory / make the folder
cd (directory_name)                  :: enter a directory // .. returns to the upper folder
ls ()                                :: returns list of files in folder
getdir ()                            :: returns directory_name
syspath ()                           :: returns system path ( == echo $PATH )
histfn (number = 5)                  :: returns list of functions called for directory, default last 5
```

System messager functionality:

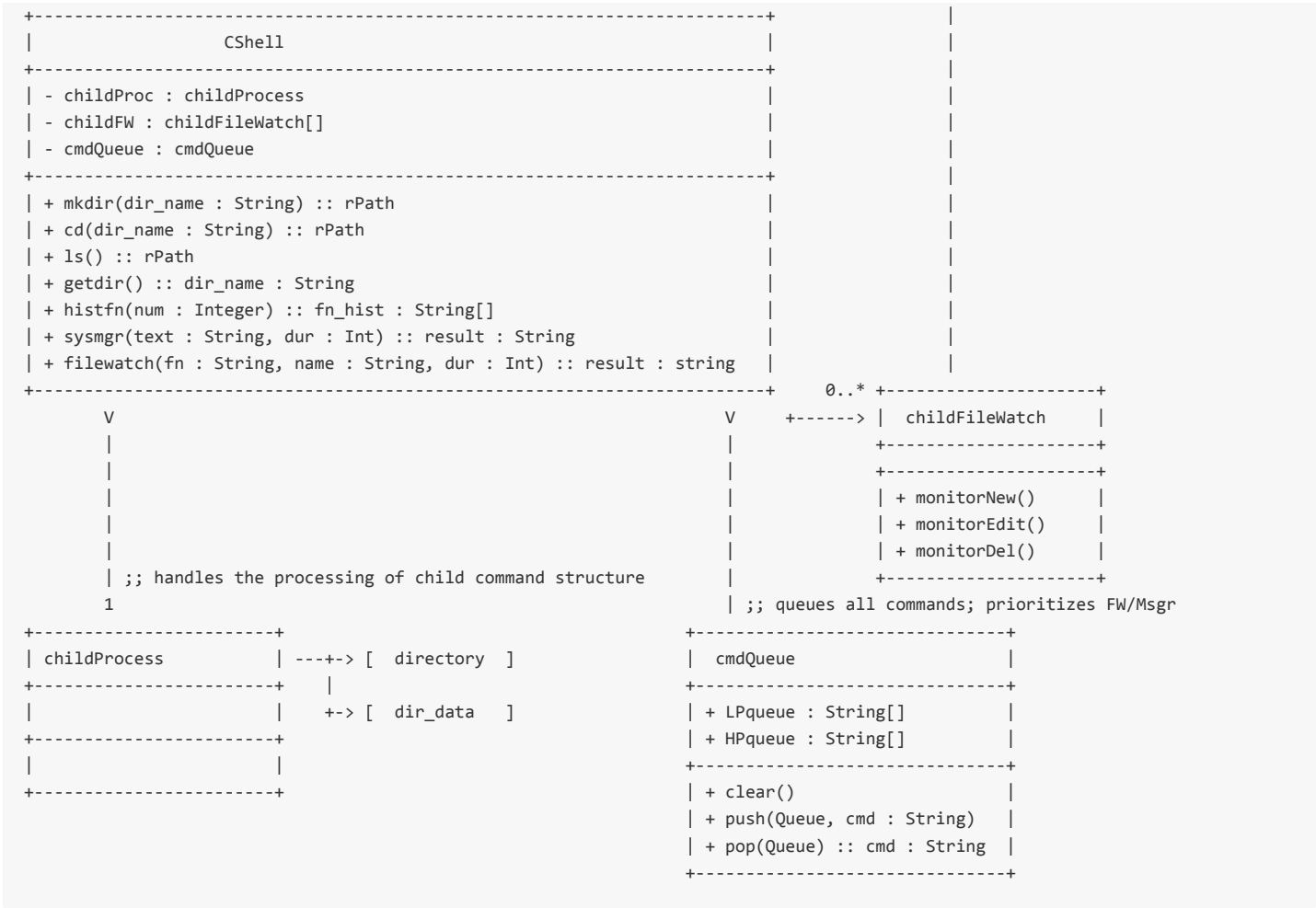
```
sysmgr (string, duration)            :: returns string after a certain duration
```

File watcher functionality:

```
filewatch (function, name, dur, action)      :: monitors for operation completed on file system for duration
```

UML Class Diagram

```
                                  +--------------------+
                                  |     directory      | ; stores path-relevant information
          +------------+          +--------------------+
          |   rPath    | |------------ | +dir_name : String |
          +------------+          +--------------------+
          | +directory | ;; goal: this object holds directory metadata,
          | +dir_data  | ;; as well as holds the files and information listed
          +------------+
          |         T                            +----------+ ; stores data at the path - incl. files, metadata
          |         +----------------------------- | dir_data | <-------------------------+
          V                                        +----------+                           |
+-------------------------------------------------------------------------+                |
|                   RShell                                                |                |
|-------------------------------------------------------------------------|                |
| - rPath                                                                 |                |
+-------------------------------------------------------------------------+                |
| + cd(dir_name : String) :: rPath                                        |                |
| + ls() :: rPath                                                         |                |
| + getdir() :: dir_name : String                                         |                |
| + histfn(num) :: fn_hist : String[]                                     |                |
| + get_jobs() :: arr_jobs : String[]                                     |                |
| + kill(signal : Int, job : String) :: result : Int                      |                |
| + sysmgr(text : String, dur : Int) :: result : String                   |                |
| + filewatch(fn : String, name : String, dur : Int, action) :: result    |                |
+-------------------------------------------------------------------------+                |
        V                                                                                  |
        |                                                                                  |
        | ;; handles the processing of each command in RShell                             |
        | ;; require revision to functional roles                                         |
```



User Stories

As a user,

```
I would like to be able to enter and leave any file in my file directory.
I would like to see what files are in what directory.
I would like to look at the history of my functions.
I would like to see if a file in a location is created, edited, or deleted.
I would like to send a message, to be returned to me at a given time.
```

As a developer,

```
I would like to run multiple filewatches at the same time, across multiple locations.
I would like to be alerted of changes in files when I'm working on other programs.
```

As a software company,

```
We would like to use a quick program that can efficiently with a multitude of filewatchers at once.
We would like to use a secure, safe application that protects itself from users, in both input and data.
```

Use Cases

Running a generic shell operation:

1. User sends request to RShell.
2. RShell communicates to CShell identifying task.

3. CShell places task into LPQueue (low-priority queue).
4. CShell pops task off of LBQueue, once childProcess is empty.
5. CShell passes task to childProcess.
6. childProcess processes task.
7. childProcess returns result to CShell.
8. CShell gets result from childProcess, returns to RShell.
9. RShell returns result to user.

Errors and Mitigations:

- 1.1 Bad data is sent to RShell
  - RShell returns error notification

Running a FileWatch or SysMgr task:

1. User sends request to RShell.
2. RShell communicates to CShell identifying task.
3. CShell places task into HPQueue (high-priority queue).
4. CShell pops task off of HPQueue.
5. CShell creates a childFileWatch to handle task.
6. childFileWatch waits until time assigned had passed.
7. childFileWatch returns result to CShell, terminates.
8. CShell returns result to RShell.
9. RShell returns result to user.

Errors and Mitigations:

- 1.1 Bad data is sent to RShell
  - RShell returns error notification
- 6.1 For FileWatcher, childFileWatch finds change in file before alloted time had passed
  - RShell returns exception found, and data collected - childFileWatch is terminated

Other References

- Class vs Module
- Ruby Exception Hierarchy
- Errno Documentation
- GetoptLong
- Regexp
- Anti-patterns