Development Diary

|  |  |  |  |
| --- | --- | --- | --- |
| **Date and time** | **Summary** | **Issues encountered** | **Notes/thoughts** |
| 22/11/2018 17:00-19:45 | Started work on stage 1. Started solution for chdir and getcwd. | Initially struggled to find out how to create system calls. |  |
| 23/11/2018 15:00-18:00 | Continued work on stage 1. Added all the required pieces for system calls to function. Continued building solution for chdir and getcwd. | Repeated cd calls weren’t adding to the CWD but were instead replacing it. |  |
| 24/11/2018 12:00-14:00 | Rewrote solution for chdir. Solution seems to work now.  Started work on stage 2. Added required pieces for the new system calls but no actual functionality. | If an attempt gets made to run some program (e.g. typing bob into the CLI, when making valid cd calls after this, the CWD will be full of unintended characters). | Does stage 1 need to check if the directories exist before making changes to myProcess()->Cwd? Seems right now just to be a glorified string appender. |
| 30/11/2018 15:00-18:00 | Fixed bug with chdir that meant the new directory after a 'cd' wouldn't be terminated correctly.  Added some more things for ls (makefile changes, ls.c) and some basic path of execution for when ls is called (when certain response codes are returned by opendir, readdir, etc.). | Initially did not understand the issue causing the chdir but, but soon found (with Wayne's help) that the safestrcpy method was not doing any terminating. | I currently do not know how best to get the current directory for when 'ls' is called without a directory specified. I also do not know how best to handle creating a directory descriptor. |
| 01/12/2018 15:00-15:30 | Added in functionality to support ‘cd ..’ to go up a directory. | None. | Could possibly implement something like cd ‘../../’ another time. |
| 06/12/2018 16:00-18:00 | Created system for getting a directoryDescriptor when calling opendir. | None. | Using a custom type ‘FileReference’ and use an array of them for the solution to this issue. FileReference is a struct which holds an int for a FileDescriptor and a File\* for the file itself. |
| 07/12/2018 15:00-18:00 | Started writing functionality to write the file data into the directory entry buffer in readdir. | Although I could get the sub-directory file itself, I don’t know how to get the files that are inside. |  |
| 10/12/2018 11:15-13:00 | Completed functionality for writing data into the directory entry buffer in readdir. Completed consumption of readdir in ls.c. | Had some issues with the formatting of the file information, but I resolved them. |  |
| 10/12/2018 14:30-17:06 | Started work on stage 3. Added some code to handle when a third argument is passed in. Added printing of additional directory entry information if ‘-l’ arg is present.  Added methods for parsing the attributes, dates, and times into a better format.  Started stage 4. | Took some time to properly format the dates, times, and attributes. | Can call ls from root directory with ‘usrbin/ls’. |
| 14/12/2018 15:00-16:45 | Continued work on stage 4. Can handle the root directory and list the files in the root directory. | All the listing occurs in the opendir system call. Need to be able to go through the chain of methods (opendir -> readdir -> closedir). |  |
| 15/12/2018 18:00-19:30 | Made changes to stage 4 solution to address the issues from the last time work was done. ‘ls’ on root directory now follows the standard chain of methods. | Because the solution implemented doesn’t seem so good, interesting decisions were made to make the solution work. E.g. putting loop variables outside methods in file.c so when readdir gets called multiple times for the root directory, the reading can resume. | After looking around in the code, a better solution to my table of custom struct FileReference, would be to implement a solution similar to:  static int fdalloc(File\* f); |
| 16/12/2018 9:30-13:15 | Completed stage 4. Confirmed ‘-l’ works for root dir as well.  Replaced existing solution for getting and setting directory descriptors with a solution similar to:  static int fdalloc(File\* f).  Fixed an issue in Stages 3 and 4 that was preventing ls <dir> from working properly.  Fixed an issue in Stage 4 that was preventing ls <dir> from working properly when called from the root directory.  Started work on stage 5. Added recognition of ‘-R’ arg. Re-wrote method for working out whether different flags are present in calls to ls. Re-wrote a lot of ls.c’s main method. Moved calls to readdir into a method so it can be called recursively. | Implementation of stage 5 solution doesn’t work right now. |  |