Sprint 1 Retrospective

Team YFFS

Team 22
Brian VerVaet, Max Molnar, Sam Spencer, Wyatt Dahlenburg, Zachary Kent

What Went Well?

- As a user, I would like to be able to create a YFFS filesystem.
  Completed By: Zachary Kent

Task Description	Actual Time	Comments
Document and test current creation functionality.	3	Was able to complete documentation for yffs-create.
Reorganize all code and create a Makefile that allows for clean creation of more tools.	2	YFFS now builds using ‘make install / uninstall’.
Create an independent tool to create the files	8	Developed yffs-create tool to create our filesystem.
Debug and test yffs-create.	5	Thoroughly tested yffs-create in a range of scenarios.

- As a user, I would like to be able to write and add files to YFFS.
  Completed By: Sam & Max

Task Description	Actual Time	Comments
Rewrite write functions so that they are atomic	2	Yffs-edit successfully adds or appends files with thread safety.
Implement write permissions in file data structure and algorithm	4	Implemented ownership permissions on both write functions and read functions.
Debug and test yffs-edit	1	Checks if either file exists and also if they are legitimate, as well as more error catching.
Create an independent tool to add files to YFFS	2	Consumes the original file once it is added.
Debug and test yffs-add.	3	Tested and debugged add, including both of its operations.

- As a user, I would like to be able read files stored in YFFS.
  Completed By: Sam & Max

Task Description	Actual Time	Comments
Rewrite read functions so that they are atomic	2	Used mutexes to create thread safety.
Implement read permissions in file data structure and algorithm	3	Implemented ownership permissions on both write functions and read functions.
Create an independent tool to list files (yffs-cat)	3	Reads the file specified, and outputs the contents of the file if (a) the file exists, and (b) the file is not empty.
Debug and test yffs-cat.	1	Debugged and tested thoroughly with a wide range of files and contents.

- As a user, I would like to be able to remove files from YFFS.
  Completed By: Brian

Task Description	Actual Time	Comments
Research atomic functionality / thread safety Rewrite remove function to be thread safe	4	Used mutexes to keep thread safety when using yffs-rm and other yffs tools.
Create an independent tool to remove files (yffs-rm)	3	Tool builds successfully with YFFS's Makefile and functions similarly to Linux rm command.
Implement editing permissions into algorithm	2	Ownership permissions for remove were implemented.
Debug and test yffs-rm.	1	Tested remove across a number of scenarios with minimal error.

- As a user, I would like to be able to list all files stored in YFFS.
  Completed By: Brian & Sam

Task Description	Actual Time	Comments
Create an independent tool to list files (yffs-ls)	3	Created tool to list files contained in the file system.
Add permissions to listing algorithm	1	Implemented ownership permissions on both write functions and read functions.
Prepare for the obfuscation interface	0.5	The listing algorithm used is basic, and should not break, assuming appropriate changes are made to the filename retrieval function.
Add functionality to export file list	3	Output of list can be exported with additional argument passed.
Debug and test yffs-ls	1	ls was debugged and is functioning well.

- As a user, I would like to have dependable data in the event of power failure.
  Completed By: Wyatt

Task Description	Actual Time	Comments
Research normal power loss situations	2	Decided there were three ways to exhibit power failure and did a small writeup at https://github.com/scspencer/yffs/blob/master/project_resources/crash/Readme
Setup raspberry pi with YFFS	2	Flashed raspbian image to raspberry pi and configured it. to run YFFS. A reboot switch was created to demonstrate a physical power loss.
Write program to simulate power loss.	3	Created bash script under the project_resources/crash folder that would reboot in the middle of a write operation. The expected output would be that the file is not written; demonstrating fault-tolerance.
Create power spikes and loss events in a program and test the fault-tolerant file system	1	Power loss situations were demonstrated by shorting the pi and by code. All situations worked and proved fault-tolerance.

What Didn’t Go Well?

Task Description	Actual Time	Comments
Collaboration using Git and Github	3	Ran into a number of issues getting everyone accustomed to using GitHub in a group environment. Successfully merging branches was a slight issue.
Unable to test power spike	1	We were never provided with a disposable raspberry pi to break. A high voltage device would be attached to the gpio pins to attempt to create a power surge.

How Could We Improve?

This sprint was the first time that our team went through the development process together. Overall the experience went smoothly, with each partner contributing in a meaningful way. However, there is always room to improve and that is what we look to do during the next sprint. While our team did a good job of communicating our intentions throughout the development process, much of the work was done close to the date of the first demo. This time around we are planning to spend additional time working on our requirements throughout the sprint, preventing any last minute work. With our user stories, we feel that using a fewer number of highly detailed user stories allows for team member specialization in certain more difficult parts of the development process. With these things in mind, our team will be able to be much more effective in sprint two.