CS450 Fall 2019
Programming Assignment 4
Jingwen Pickens
Anthony Pollizi

Test Programs

These test cases were selected to show what is expected to happen and what actually happens when small files directory and small files were created, as well as how open(), read() and write() system calls behave when small files are involved.

- **Test overflow (1)** is designed to verify that our implementation is able to catch write() overflow of a small file and throws an error message.
- **Test regfiles (2)** is designed to verify that open(), read(), and write() system calls still behave the same for regular file directories and regular files.
- **Test sfile (3)** is designed to verify that (1) mkSFdir() is able to create small file directories and small files that fit into an inode are able to be created inside a small file directory. (2) open(), read() , and write() system calls behaves normally for small files.
- **Test dirs (4)** is designed to verify that (1) both small file subdirectories and regular file subdirectories can be created inside of a small file directory; (2) small file subdirectories can be created inside of a regular file directory.
- **Test unlink (5)** is designed to verify that small files can be removed by unlink().

These below test cases will be run in the running XV6 shell after executing the command **tests**. Test names are in bold.

1. **Test overflow (1)** creates small file directory *sdir* using mkSFdir(). It then creates a small file named *sfile* inside *sdir* by calling open() with O_CREATE and O_RDWR flags. Then, a string of size of 100, which is larger than the small file size limit of 52 bytes, is written into the small file *sfile*. This test is expected to successfully create a small file *sfile* inside small file directory *sdir* and a write overflow should occur. An error message “Write would overflow a small file”. Another error message “error: write to *sfile* failed” should be printed out. Finally “Test overflow (1) passed” should be printed out.

   Upon completion of the entire tests.c program, an execution of command **ls** should show *sdir* as type 4. Command **ls sdir** should show *sfile* as type 5 with a size of 0 since the write() failed. The actual test output matches with the expected output.

2. **Test regfiles (2)** creates a regular file directory *regulardir* using mkdir(). It then creates a regular file named *regularfile* inside *regulardir* by calling open() with O_CREATE and

O_RDWR flags. Then a string of size 100 is written into *sfile1* by calling write(), followed by closing the file descriptor fd. The file *regularfile* is then opened again this time by calling open() with O_RDONLY flag and the content in the file is read by calling read() and the content is printed out.

This test is expected to successfully create a regular file directory and a regular file, open and write into that file and close it. It should then successfully open the file again and read content from the file and print it out. The output should match what was written into it. Finally "Test regfiles (2) passed" should be printed out.

Upon completion of the entire tests.c program, an execution of command **ls** should show *regulardir* as type 1. Command **ls *regulardir*** should show *regularfile* as type 2 with a size of 100. The actual test output matches with the expected output.

3. **Test sfiles (3)** creates small file directory *sfiledirtest* using mkSFdir(). It then creates a small file named *sfile1* inside *sfiledirtest* by calling open() with O_CREATE and O_RDWR flags. Then a string of size 40, which is no larger than the small file size limit 52 bytes, is written into *sfile1* by calling write(), followed by closing the file descriptor fd. The file *sfile1* is then opened again this time by calling open() with O_RDONLY flag and the content in the file is read by calling read() and the content is printed out.

   This test is expected to successfully create the small file directory, create a small file, open and write into that file and close it. It should then successfully open the file again and read content from the file and print it out. The output should match what was written into it. Finally "Test sfiles (3) passed" should be printed out.

   Upon completion of the entire tests.c program, an execution of command **ls** should show *sfiledirtest* as type 4. Command **ls *sfiledirtest*** should show *sfile1* as type 5 with a size of 40. The actual test output matches with the expected output.

4. **Test dirs (4)** makes small files subdirectories *isdir* and regular file subdirectories *idir* inside of a small file directory *sdir* using mkSFdir("sdir/isdir") and mkdir("sdir/idir") This test also makes a small file subdirectory *sdir* inside of a regular file directory *dir* using mkSFdir("dir/sdir").

   This test is expected to create all subdirectories successfully and "Test dirs (4) passed" should be printed out. Upon completion of the entire tests.c program, an execution of command **ls sdir** should show small file subdirectory *isdir as type 4 and* regular file subdirectory *idir as type 1*. Command **ls dir** should show small file subdirectory *sdir* as type 4. The actual test output matches with the expected output.

5. **Test unlink (5)** creates a small file *sfiletodelete* inside the small file directory *sdir* and then it removes it by calling unlink("sdir/sfiletodelete"). This test is expected to successfully remove the small file and "Test unlink (5) passed" should be printed out. Upon completion of the entire tests.c program, an execution of command **ls sdir** should not show *sfiletodelete* in the *sdir* directory. The actual test output matches with the expected output.