TEST CASES

Group 03
Henri-Philippe Marceau
Kaelan Renault
Kaitlynn Anderson

TESTS1.c

TESTS1 attempts to test the read, write, create, close and unlink functionality for Task 1. It uses a file pointer (fptr) to a file named Imaginary. The process goes as follows:

First, it attempts to open the non-existent file using the open sysCall, which then throws -1, indicating that the open has failed.

Second, it attempts to create the non-existent file using the creat sysCall, which should then create the file Imaginary.

Third, it tries to open Imaginary with the sysCall open. This should add a second reference to the file.

Fourth, It tries to create another instance of Imaginary, using the creat system call. This only adds another reference to the file.

It then attempts to open the max amount of files. Immediately followed by an attempt to close all references to the file and unlink the last one. Deleting Imaginary.

It then attempts to open Imaginary again to see if it has been removed.

If all goes well, the string “TEST 1 - PASS” is printed

If at any point an error occurs, -1 is printed for that variable

TESTS1.c output obtained via ‘nachos -x TESTS1.coff’ :

```
nachos 5.0j initializing... config interrupt timer processor
console user-check grader
Testing the console device. Typed characters
will be echoed until q is typed.
q
Attempt at opening non-existent file:
     File Descriptor: -1

Attempt at creating non-existent file:
     File Descriptor: 15

Attempt at opening existing file:
     File Descriptor: 14

Attempt at creating existing file:
     File Descriptor: 13


Open max amount of files

File Descriptor: 12
File Descriptor: 11
File Descriptor: 10
File Descriptor: 9
File Descriptor: 8
File Descriptor: 7
File Descriptor: 6
File Descriptor: 5
File Descriptor: 4
File Descriptor: 3
File Descriptor: 2

 Close remaining files
Closing File Descriptor: 2
Closing File Descriptor: 3
Closing File Descriptor: 4
Closing File Descriptor: 5
Closing File Descriptor: 6
Closing File Descriptor: 7
Closing File Descriptor: 8
Closing File Descriptor: 9
Closing File Descriptor: 10
Closing File Descriptor: 11
Closing File Descriptor: 12
Closing File Descriptor: 13
Closing File Descriptor: 14
unlinking: Imaginary
Test 1 - PASS
Machine halting!
```

TESTS2.c

TESTS2 attempts to test using the create, close, and unlink calls using an invalid file name, . It uses a file pointer (fptr) to a file named ‘ ‘ . The process goes as follows:

It first attempts to create the file using the variable fd and the sysCall create, which should fail and return and print -1

It then attempts to close the file, using the variable close and the sysCall close, which should fail and return and print -1

It then attempts to unlink the file, using the variable unlk and the sysCall unlink, which should fail and return and print -1

TESTS2.c output obtained via ‘nachos -x TESTS2.coff’ :

```
nachos 5.0j initializing... config interrupt timer processor
console user-check grader
Testing the console device. Typed characters
will be echoed until q is typed.
q
Attempt at creating a file with an invalid file name:
     File Descriptor: -1
Attempt to close invalid File Descriptor:
     Close Return Code: -1
Attempt to unlink non-exsitent file:
     Unlink Return Code: -1
Machine halting!
```

readFile.c

readFile.c attempts to read the contents of the file ‘text.txt’. It then displays the contents of ‘text.txt’ which reads “Hello World!”.

readFile.c output is obtained via ‘nachos -x readFile.coff’:

```
nachos 5.0j initializing... config interrupt timer processor console
user-check grader
Testing the console device. Typed characters
will be echoed until q is typed.
q
File descriptor: 15
Bytes read: 13
```

```
Contents of text.txt:
Hello World!

Machine halting!
```

writeFile.c

writeFile.c attempts to write “HELLO WORLD!” to file “test.txt”. If “test.txt” doesn’t exist it is created before being written to. Afterwards the File Descriptor and bytes written are displayed.

writeFile.c output is obtained via ‘nachos -x writeFile.coff’:

```
nachos 5.0j initializing... config interrupt timer processor console
user-check grader
Testing the console device. Typed characters
will be echoed until q is typed.
q
File descriptor: 15
Bytes written: 13
Machine halting!
```

Contents of “test.txt”:

```
HELLO WORLD!
```

TESTS3.c

TESTS3 attempts to read and write to and from invalid file descriptors. It then displays -1 for both amount of bytes read and written.

TESTS3.c output obtained via ‘nachos -x TESTS3.coff -d aj’ :

```
nachos 5.0j initializing... config interrupt timer processor console
user-check grader
Testing the console device. Typed characters
will be echoed until q is typed.
q
Bytes read: -1
Bytes written: -1
Machine halting!
```

TESTS4.c

TESTS4 attempts to test the exec sysCall for Task 3. It uses a file pointer (fptr) to a file named hello.coff The process goes as follows:

It calls the exec syscall with arguments fptr, 0 and null. It then joins the hello.coff process.

TESTS4.c output obtained via ‘nachos -x TESTS4.coff -d a’ :

```
nachos 5.0j initializing... config interrupt timer processor
console user-check grader
Testing the console device. Typed characters
will be echoed until q is typed.
q
UserProcess.load("TESTS4.coff")
     initializing .text section (3 pages)
     initializing .rdata section (1 pages)
     initializing .data section (0 pages)
     initializing .bss section (1 pages)
Exec hello.coff
UserProcess.load("hello.coff")
     initializing .text section (3 pages)
     initializing .rdata section (1 pages)
     initializing .data section (0 pages)
     initializing .bss section (1 pages)
Hello, World!
Machine halting!
```

TESTS5.c

TESTS5 attempts to test the join sysCall, for Task 3. It uses two file pointers called fptrA and fptrB which joins to two files joinA.coff and joinB.coff respectively. The process goes as follows:

It calls exec on joinA.coff, then calls exec on joinB.coff passing joinA’s PID as an argument. It then joins both joinA and joinB. At this point, joinB attempts to join joinA, a non-child process, this returns -1. Next, joinA exits with an exit code of 23 and joinB exits with an exit code of 127. TESTS5 then prints out each processes’ PID and return codes and finishes.

TESTS5.c output obtained via ‘nachos -x TESTS5.coff -d a’ :

```
nachos 5.0j initializing... config interrupt timer processor console
user-check grader
Testing the console device. Typed characters
will be echoed until q is typed.
q
UserProcess.load("TESTS5.coff")
	initializing .text section (3 pages)
	initializing .rdata section (1 pages)
	initializing .data section (0 pages)
	initializing .bss section (1 pages)
UserProcess.load("joinA.coff")
	initializing .text section (1 pages)
	initializing .data section (0 pages)
	initializing .bss section (0 pages)
UserProcess.load("joinB.coff")
	initializing .text section (3 pages)
	initializing .rdata section (1 pages)
	initializing .data section (0 pages)
	initializing .bss section (1 pages)
B attempting to join non-child process A
Join attempt's return code: -1
A's PID: 1
B's PID: 2
A's Return Code: 23
B's Return Code: 127
Machine halting!
```

TESTS6.c

TESTS6 attempts to test exec by calling exec with a series of invalid arguments. It uses two file pointers called fptr and fptr2 to reference Imaginary and hello.c respectively. The process goes as follows:

First it attempts to exec Imaginary. This should return the error code -1, which then gets printed to the console.

Then it attempts to exec hello.c using a negative argument for argc, and attempts to execute. This should return the error code -1, which then gets printed to the console.
It then attempts to execute hello.c using a mismatched argc and argv. This should also fail and return the error code -1, which then gets printed to the console.

TESTS6.c output obtained via ‘nachos -x TESTS6.coff’ :

```
nachos 5.0j initializing... config interrupt timer processor console
user-check grader
Testing the console device. Typed characters
will be echoed until q is typed.
q
Trying to Exec non existent file: -1
Trying to Exec with negative argc: -1
Trying to Exec with mismatched argc, argv: -1
Machine halting!
```

UserProcess selfTest()

The UserProcess selfTest() method tests the functionality of task 2’s implementation. It focuses on testing the rigorousness of the readVirtualMemory and writeVirtualMemory methods. The method goes as follows:

It starts by writing a message, “SUCCESS”, to memory and attempting to read that very same message from memory.

Afterwards, it tries to write more than a pages worth of bytes to memory. This case is handled by the writeVirtualMemory method which writes the overflow to the next virtual page. The overflow in question is 4 bytes and contains the message “GOOD”. The amount of bytes written is then displayed.

At this point, the method attempts to read more than pages worth of bytes from memory. This case is handled by readVirtualMemory which reads the overflow from the next virtual page. The overflow in question are the very same 4 bytes that were written prior to the readVirtualMemory call. The last 4 bytes containing the message “GOOD” are displayed along with the amount of bytes read.

Next, to truly confirm UserProcess’ ability to properly access memory across multiple virtual pages. The selfTest() method calls readVirtualMemory() to read the first 4 bytes of the second virtual page, vpn 1. If this implementation of nachos properly accesses memory, these 4 bytes should contain the message “GOOD”.

This output can be obtained via ‘nachos -d j’:

```
nachos 5.0j initializing... config interrupt timer processor console
user-check grader
Read Write Test: SUCCESS
```

```
Bytes Written: 1028
Write OverFlow Test: GOOD
Bytes Read: 1028
Read OverFlow Test: GOOD
OverFlow Test: GOOD
Testing the console device. Typed characters
will be echoed until q is typed.
q
Machine halting!
```