Computer Architecture and Technology Area

Universidad Carlos III de Madrid

OPERATING SYSTEMS

Lab 1. System calls

BACHELOR'S DEGREE IN COMPUTER SCIENCE AND ENGINEERING

Year 2016/2017

Skúli Arnarsson – 100369706

Table of contents

Table of Contents

Assignment 1: mycat.c

Code description:

The program takes a path to a file as an argument. If no argument is provided the program returns an error and an error message that not enough arguments are provided. The program allocates a buffer of a defined size of 1024 bytes, if the allocation fails the error is handled. The program then opens the file that is passed as an argument, saves the returned file descriptor and handles the error if the file was not opened correctly. Next the program reads the contents of the file to the previously allocated buffer and handles the error if there was problem reading to it. Then the contents of the buffer are written to the standard output and the file descriptor is closed and of course there is error handling for both operations.

Test cases:

Usage: ./mycat <path_to_input_file>

Test case 1:

$ ./mycat p1_tests/f1.txt

Expected output:					Obtained output:				
Nombre1	V	32	09834320	24500	Nombre1	V	32	09834320	24500
Nombre2	M	35	32478973	27456	Nombre2	M	35	32478973	27456
Nombre3	V	53	98435834	45000	Nombre3	V	53	98435834	45000

Test Passed.

Test case 2:

$ ./mycat p1_tests/Lorem\ Ipsum.txt

Clarification: Here I created a file named Lorem Ipsum.txt and added the text below to it

Expected output	Obtained output:
"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.“	"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Test Passed.

Test case 3:

$ ./mycat p1_tests/not_exists.txt

Expected output	Obtained output:
Error opening file Returned -1	Error opening file Returned -1

Test passed

Test case 4:

$ ./mycat p1_tests/

Expected output	Obtained output:
Not enough arguments Returned -1	Not enough arguments Returned -1

Test passed

Requirements:	**Satisfied**
The program must show the whole contents of the file.	✓
The program must return -1 if no argument was passed.	✓
The program must return -1 if there was an error when opening the file (e.g. the file does not exist).	✓

Assignment 2: myls.c

Code description:

The program has two usages, either it takes no arguments and lists the entire contents of the current directory with each entry on a separate line or it takes one argument, a valid path to a directory and works in the same manner with the directory provided as an argument instead. If more than one arguments are provided the program only lists the contents of the first one and ignores the others.

If the program is not able to open the specified directory the error is handled and returns a value of -1. A pointer is set to the first entry in the directory by performing a directory read on the pointer to the directory (which is either set to the current working directory or to the specified directory provided as an argument). The program prints the name of the entry the pointer is set to and prints the name of that entry. As long as there exists another entry in the directory that has not been listed the program updates the pointer to the next entry and prints the name of that entry. Finally, the directory pointer is closed.

Test cases:

Usages:

Usage: ./myls

Usage 2: ./myls <directory>

Test case 1:

$./myls

Expected output:

List the entire contents of the current directory, like the ls command but list each entry in a separate line:

$ ls -f -1 command for reference:

Expected output (with $ ls -f -1)	Obtained output:
. .. corrector_ssoo_p1.py Makefile mycat mycat.c mycat.exe mycat.o myls myls.c myls.exe myls.o mysize mysize.c	. .. corrector_ssoo_p1.py Makefile mycat mycat.c mycat.exe mycat.o myls myls.c myls.exe myls.o mysize mysize.c

mysize.o ~$oo_p1_100369706.odt p1_tests ssoo_p1_100369706.odt ssoo_p1_100369706.zip submission ~WRL0613.tmp	mysize.o p1_tests ssoo_p1_100369706.odt ssoo_p1_100369706.zip submission ~$oo_p1_100369706.odt ~WRL0613.tmp

Test passed

Test case 2:

$ ./myls p1_tests/

Expected output (with $ ls -f -1)	Obtained output:
. .. dirA dirC f1.txt f2.txt Lorem Ipsum.txt	. .. dirA dirC f1.txt f2.txt Lorem Ipsum.txt

Test passed

Test case 3:

$ ./myls notExists/

Expected output	Obtained output:
Error opening the directory Returned -1	Error opening the directory Returned -1

Test passed

Requirements	**Satisfied**
The program must list all entries of the directory , in the order in which the call to readdir returns them, and show each entry in one line.	✓
The program must list the entries of the directory passed as parameter (usage 2), or from the current directory if no parameter was passed.	✓
The program must return -1 if an error happened while opening the directory (e.g, the directory does not exist).	✓

Assignment 3: mysize.c

Code description:

The program takes no arguments, if any are provided they are ignored. The program lists all entries in the current working directory by name, followed by a tab character (\t) and the size of the entry, each on a separate line.

If the program is not able to open the specified directory the error is handled and returns a value of -1. A pointer is set to the first entry in the directory by performing a directory read on the pointer to the current working directory. For each entry in the directory the program checks if that entry is a regular file, if it is not a regular file it is not listed by ./mysize. For each regular file the file is opened and the file descriptor is stored. If there is an error opening the file, the program indicates so and returns a value of -1. Then the size of the file is calculated, the size of the file equals the byte offset of seeking to the end of the file. If there is an error seeking to the end of the file, the program indicates so and returns a value of -1. Subsequently the program prints the name of each entry and the previously calculated size. Finally, the program closes the pointer to the directory and checks if it closes correctly, else returns a value of -1.

Test cases:

Usage: ./mysize

Test case 1:

cd p1_pruebas/

$ ../mysize

Expected output	Obtained output:
f1.txt 87 f2.txt 87	f1.txt 87 f2.txt 87

Test case 2:

cd p1_llamadas

$ ../mysize

Expected output	Obtained output:
corrector_ssoo_p1.py 5172 Makefile 308 mycat 8773 mycat.c 1489 mycat.exe 31012 mycat.o 2376 myls 8842 myls.c 1564 myls.exe 34475	corrector_ssoo_p1.py 5172 Makefile 308 mycat 8773 mycat.c 1489 mycat.exe 31012 mycat.o 2376 myls 8842 myls.c 1564 myls.exe 34475

myls.o 2256 mysize 8996 mysize.c 2266 mysize.o 2816 ssoo_p1_100369706.odt 32252 ssoo_p1_100369706.zip 5661 ~$oo_p1_100369706.odt 162 ~WRL0613.tmp 24478	myls.o 2256 mysize 8996 mysize.c 2266 mysize.o 2816 ssoo_p1_100369706.odt 32252 ssoo_p1_100369706.zip 5661 ~$oo_p1_100369706.odt 162 ~WRL0613.tmp 24478

Test passed

Test case 3:

cd p1_pruebas/

$ ../mysize notExists/

Expected output: Print the current directory

Expected output	Obtained output:
f1.txt 87 f2.txt 87	f1.txt 87 f2.txt 87

Test passed

Requirements	**Satisfied**
The program must show the name and size of all the regular files of the directory, in the order in which the call readir returns them, and showing the data of each file in one line.	✓
The program will only show the data from regular files.	✓
The program will show the data using the following format:<name><tab><size>	✓
The program must return -1 if there was an error when opening the file.	✓

Conclusion

The main problems I had were not serious and I easily figured them out with a quick Google search. I had trouble remembering the arguments required for a system call and what value each of them returned, so I used http://codewiki.wikidot.com for reference. I had the most trouble with the "open" system call and figuring out which oflag parameter to use.

I felt that I learned a lot about the uses of system calls and how they work when working on this assignment whereas I felt completely lost before completing it.