

CSE344 – System Programming

HW1

Report

 Seniha Sena Topkaya

 131044020

Part1: appendMeMore

First of all, I made the program work for the arguments given as relative and absolute path. After that, I opened the file with the O_APPEND flag by default. I wrote the number of characters specified in the argument to the file. If the 'x' argument was given, this time I opened the file without using the O_APPEND flag, and first using the lseek function with SEEK_END, I moved the file cursor to the end and then wrote a character to the file. Finally, I closed the file and deallocate the memory I was using.

Output:

- If command line arguments are invalid, print usage information and exit:

I check all the arguments and if they are invalid then I call printUsage function to print valid input format.

```
pi@raspberrypi:~/Desktop/topkaya_seniha_sena_131044020 $ ./appendMeMore
Usage: ./appendMeMore filename num-bytes [x]
pi@raspberrypi:~/Desktop/topkaya_seniha_sena_131044020 $
```

- Relative or Absolute Path – Both of them work

```
pi@raspberrypi:~/Desktop/topkaya_seniha_sena_131044020 $ ./appendMeMore /home/pi/Desktop/inp.txt 10
pi@raspberrypi:~/Desktop/topkaya_seniha_sena_131044020 $
```

- Results:

```
pi@raspberrypi:~/Desktop/topkaya_seniha_sena_131044020 $ ./appendMeMore f1 1000000 & ./appendMeMore f1 1000000
[1] 5075
pi@raspberrypi:~/Desktop/topkaya_seniha_sena_131044020 $ ./appendMeMore f2 1000000 x & ./appendMeMore f2 1000000 x
[2] 5081
open file: File exists
[1] Done ./appendMeMore f1 1000000
pi@raspberrypi:~/Desktop/topkaya_seniha_sena_131044020 $ ls -l
total 3028
-rwxr-xr-x 1 pi pi 15920 Mar 27 17:12 appendMeMore
-rw-r--r-- 1 pi pi 4101 Mar 27 17:05 appendMeMore.c
-rw-r--r-- 1 pi pi 3910 Mar 27 17:12 appendMeMore.o
-rw-r--r-- 1 pi pi 2000000 Mar 27 17:12 f1
-rw-r--r-- 1 pi pi 1000000 Mar 27 17:12 f2
-rw-r--r-- 1 pi pi 98 Mar 27 11:15 inp2.txt
-rw-r--r-- 1 pi pi 111 Mar 27 11:15 inp.txt
-rw-r--r-- 1 pi pi 470 Mar 27 12:31 makefile
-rwxr-xr-x 1 pi pi 15904 Mar 27 17:12 mydup
-rw-r--r-- 1 pi pi 3341 Mar 27 17:02 mydup.c
-rw-r--r-- 1 pi pi 3980 Mar 27 17:12 mydup.o
-rwxr-xr-x 1 pi pi 15772 Mar 27 17:12 part3
-rw-r--r-- 1 pi pi 1380 Mar 27 17:11 part3.c
-rw-r--r-- 1 pi pi 2340 Mar 27 17:12 part3.o
-rw-r--r-- 1 pi pi 292 Mar 27 13:33 test.txt
[2]+ Done ./appendMeMore f2 1000000 x
pi@raspberrypi:~/Desktop/topkaya_seniha_sena_131044020 $
```

```

pi@raspberrypi:~/Desktop/topkaya_seniha_sena_131044020 $ ./appendMeMore f1 1000000 & ./appendMeMore f1 1000000
pi@raspberrypi:~/Desktop/topkaya_seniha_sena_131044020 $ ./appendMeMore f2 1000000 x & ./appendMeMore f2 1000000 x
[2] 2629
[1] Done
./appendMeMore f1 1000000
pi@raspberrypi:~/Desktop/topkaya_seniha_sena_131044020 $ ls -l
total 3988
-rwxr-xr-x 1 pi pi 15884 Mar 28 16:48 appendMeMore
-rw-r--r-- 1 pi pi 2987 Mar 27 17:57 appendMeMore.c
-rw-r--r-- 1 pi pi 3276 Mar 28 16:48 appendMeMore.o
-rw-r--r-- 1 pi pi 2000000 Mar 28 16:48 f1
-rw-r--r-- 1 pi pi 1999449 Mar 28 16:49 f2
-rw-r--r-- 1 pi pi 476 Mar 27 12:31 makefile
-rwxr-xr-x 1 pi pi 15904 Mar 28 16:48 mydup
-rw-r--r-- 1 pi pi 3373 Mar 28 15:52 mydup.c
-rw-r--r-- 1 pi pi 3992 Mar 28 16:48 mydup.o
-rwxr-xr-x 1 pi pi 15772 Mar 28 16:48 part3
-rw-r--r-- 1 pi pi 1406 Mar 27 18:19 part3.c
-rw-r--r-- 1 pi pi 2344 Mar 28 16:48 part3.o
[2]+ Done
./appendMeMore f2 1000000 x
pi@raspberrypi:~/Desktop/topkaya_seniha_sena_131044020 $

```

As a result, both file sizes and file contents are different. **When I run the program over and over, the sizes change, but the size of the file I always use lseek is smaller than the size of the other file.**

Since the cursor moves automatically when I use **O_APPEND**, **both program instances can write to the file without waiting.** When I don't use O_APPEND, an instance of a program writes a character to the file and moves the cursor to the end of the file with lseek. But while lseek is not done yet, the other program instance writes a character to the file. As a result, some characters are written in the same place so it's overwriting. Therefore, when the program runs with the x argument (using lseek+write instead of O_APPEND), the file size is smaller than expected.

Part2: mydup

First of all, I used the fcntl function with F_DUPFD while implementing the dup function. In the dup2 function, I first checked whether the newly created file descriptor is valid. I then checked if the old descriptor was valid using the fcntl function F_GETFL. If both descriptors are valid and both are equal, one is returned. After all the checks were done, I closed the newly created descriptor. Finally, I have duplicated the descriptor by calling the fcntl function with F_DUPFD.

While testing my mydup and mydup2 functions in the main function, I also wanted to implement **part3**. For testing, I first created strings. Then I opened a file and duplicated the file descriptor of this file with mydup. I wrote the strings to the opened file using both the old and the new file descriptor. In order to perform part3, I printed where the file cursor was on the screen. I did the same testing process to try the mydup2 function.

Output:

- I test my dup and dup2 functions to make sure if they work well. First I checked if they write something in a same file without reopen and then I checked if they share the same offset. Here are terminal result and input files:
- **Test 1:** if old file descriptor is not valid. I printed error message.
- **Test 2:** if new file descriptor is not valid. I printed error message.
- **Test 3:** if file descriptors are same and if both of them are valid. Then mydup2 function will work. For this case the output file is inp3.txt. File descriptors share the same file offset.

```
pi@raspberrypi:~/Desktop/topkaya_seniha_sena_131044020 $ ./mydup
fd offset is : 92
fdCpy offset is : 92
fd2 offset is : 76
fd3 offset is : 76
_____Test part for error handling_____
Error: : Bad file descriptor
mydup2 error (invalid oldfd) : Bad file descriptor
Error: : Bad file descriptor
mydup2 error (invalid newfd) : Bad file descriptor
fd4 offset:36
fd5 offset:36
pi@raspberrypi:~/Desktop/topkaya_seniha_sena_131044020 $
```

Test 1

Test 2

Test 3

inp.txt

```
inp.txt x inp2.txt x
1 mydup - first file descriptor --> fd
2 mydup - the copy of my first file descriptor --> fdCpy
3
```

inp2.txt

```
inp.txt x inp2.txt x
1 dup2 - mydup2 file descriptor --> fd2
2 dup2 - mydup2 file descriptor --> fd3
3
```

inp3.txt

```
mydup.c x inp3.txt x part3.c
1 dup2 - mydup2 same file descriptors
2
```

Part3: part3

As in the second part, I first opened a file and used the dup function to duplicate the file descriptor of this file. This time I wrote a test using the dup function of the unistd library. To test, I defined strings and wrote these strings to the opened file using different file descriptors. Then, to see if they share the offset, I called the lseek function with SEEK_CUR and printed the current position of the cursor on the screen.

Output:

- In this part I test dup and dup2 function. First I checked if they write something in a same file without reopen and then I checked if they share the same offset. test.txt is the output file for dup and test2.txt is output file of dup2 function.
- **Test 1:** if old file descriptor is not valid.
- **Test 2:** if new file descriptor is not valid.
- **Test 3:** if file descriptors are same and if both of them are valid. Then dup2 function will work. For this case the output file is test3.txt. File descriptors share the same file offset.

Here are terminal result and input file:

```
pi@raspberrypi:~/Desktop/topkaya_seniha_sena_131044020 $ ./part3
fd offset is : 69
fdCpy offset is : 69
fd2 offset is : 58
fd3 offset is : 58
_____Test part for error handling_____
dup2 error (invalid oldfd) : Bad file descriptor
dup2 error (invalid newfd) : Bad file descriptor
fd4 offset:29
fd5 offset29
pi@raspberrypi:~/Desktop/topkaya_seniha_sena_131044020 $
```

Test 1

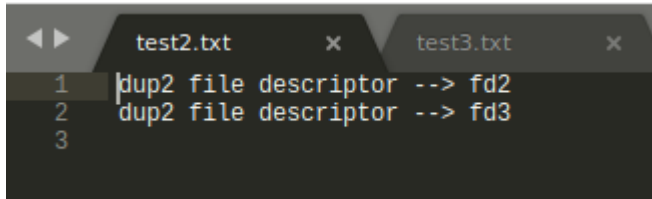
Test 2

Test 3

test.txt

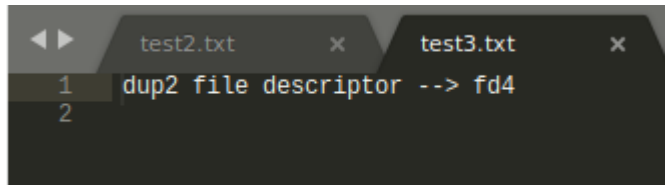
```
part3.c  x  test.txt  x
1 |first file descriptor --> fd
2 |copy of the first file descriptor --> fdCpy
3 |
```

test2.txt



```
1 dup2 file descriptor --> fd2
2 dup2 file descriptor --> fd3
3
```

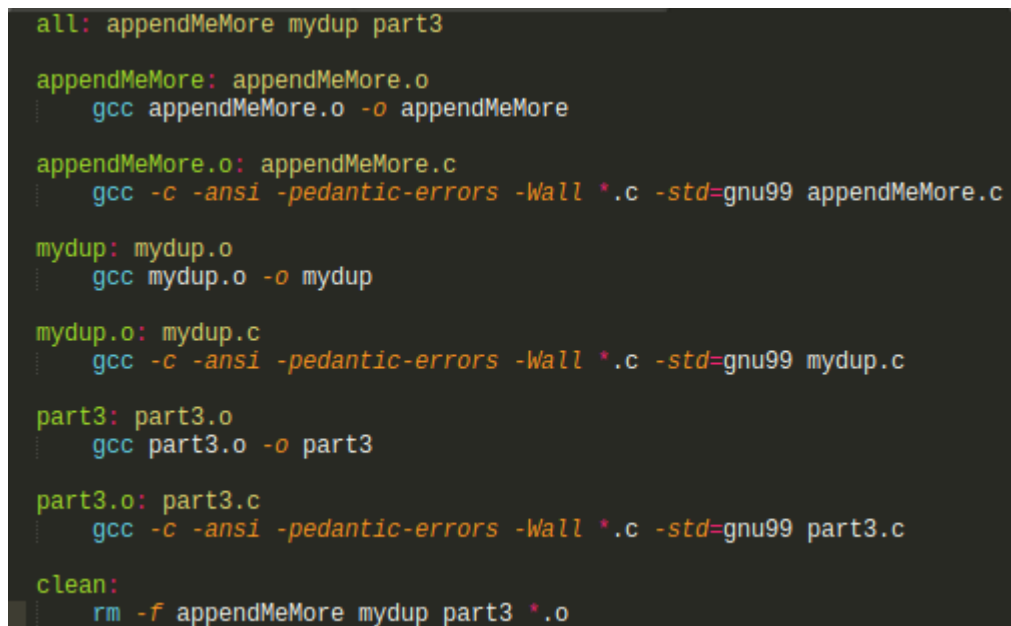
test3.txt



```
1 dup2 file descriptor --> fd4
2
```

General rules:

- Makefile with make clean that just compile the program not run for all source files.



```
all: appendMeMore mydup part3

appendMeMore: appendMeMore.o
    gcc appendMeMore.o -o appendMeMore

appendMeMore.o: appendMeMore.c
    gcc -c -ansi -pedantic-errors -Wall *.c -std=gnu99 appendMeMore.c

mydup: mydup.o
    gcc mydup.o -o mydup

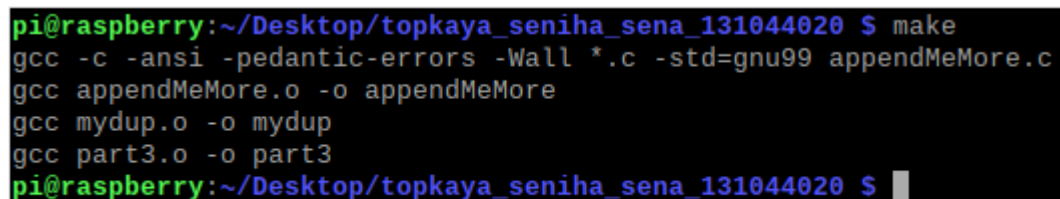
mydup.o: mydup.c
    gcc -c -ansi -pedantic-errors -Wall *.c -std=gnu99 mydup.c

part3: part3.o
    gcc part3.o -o part3

part3.o: part3.c
    gcc -c -ansi -pedantic-errors -Wall *.c -std=gnu99 part3.c

clean:
    rm -f appendMeMore mydup part3 *.o
```

- There is no compilation error



```
pi@raspberrypi:~/Desktop/topkaya_seniha_sena_131044020 $ make
gcc -c -ansi -pedantic-errors -Wall *.c -std=gnu99 appendMeMore.c
gcc appendMeMore.o -o appendMeMore
gcc mydup.o -o mydup
gcc part3.o -o part3
pi@raspberrypi:~/Desktop/topkaya_seniha_sena_131044020 $
```

End of the report