

All Practical Assignments Documentation

Teodora Kostoska 0510085

All of the code contains a lot of comments, which should explain better how the code works, so this is only to explain the main concept and the steps taken. References can be found from the .c files for Assignment 1 and Assignment 2. Link to repository where all of the programs are is at the end of this document.

Assignment 1:

The project was started by figuring out how to open and print a file one line at a time. Then I created a linked list where I could keep each line. The idea was to always add the new line as the first element of the linked list. As printing was done by starting at the beginning of the linked list the output was a reversed text.

Next I did the code for the case when there is an output file given. I began by figuring out how to write to a file, after which I just printed the elements of the linked list to standard output. As in both of the cases (only input file and input and output file) the reading of the input file is necessary, these two options are done under the same section. The output option is then checked after the lines from the input file have been added to the linked list.

After that I did the option of when there are no files given, where the program must read from the standard input. This was easy to implement as the idea is the same as reading lines from an input file, but in the space of the file there is stdin. I made it so that the reading of the standard input end when the user doesn't give any input other than enter. The section is otherwise same as in the other cases.

Lastly I did all of the additional assumptions and error checks specified in the assignment. The program is tested and works with all of the assumptions given.

Demonstrations of the program:

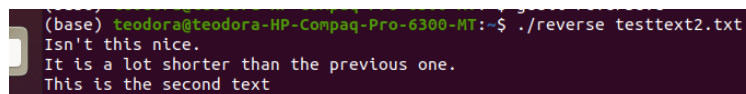
The testtext2.txt file contains the lines:

This is the second text

It is a lot shorter than the previous one.

Isn't this nice.

Output of call `./reverse testtext2.txt` (picture 1), output of call `./reverse` (picture 2), output of call `./reverse` unavailable file (picture 3), (code works for output file also, but I didn't know how to show that here) :



```
(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./reverse testtext2.txt
Isn't this nice.
It is a lot shorter than the previous one.
This is the second text
```

(Picture 1)

```

(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./reverse
Empty line or only enter ends the loop.
Giving a line from standard input
This is going to be the seconf line
Maybe one more
And this one too
This could go forever
But I'm too lazy to write more

But I'm too lazy to write more
This could go forever
And this one too
Maybe one more
This is going to be the seconf line
Giving a line from standard input
(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./reverse input2.txt output.txt

```

(Picture 2)

```

Giving a line from standard input
(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./reverse input2.txt output.txt
error: cannot open file 'input2.txt'
(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./reverse testtext2.txt output.txt

```

(Picture 3)

Assignment 2:

my-cat:

This mini project was pretty similar to the previous assignment, just a lot less lines to write. I started with making a program that opens the file and reads it, but instead of using `getline` I used `fgets`. After this I just checked that the program was done according to all of the assumptions given and did some error handling.

Examples of the program working (picture 1: Printing a single file, picture 2: printing multiple files, picture 3: no file given, picture 4: error opening file):

```

(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./my-cat testtext2.txt
Reading file: 'testtext2.txt'

This is the second text
It is a lot shorter than the previous one.
Isn't this nice.
(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./my-cat testtext2.txt testtext4.txt

```

```

(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./my-cat testtext2.txt testtext4.txt
Reading file: 'testtext2.txt'

This is the second text
It is a lot shorter than the previous one.
Isn't this nice.
Reading file: 'testtext4.txt'

this is a test of standard input
you should see this line in the output because it has words in it
this line also has words
but this one doesnt
nor does this one
(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./my-cat

```

```

(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./my-cat
No files given.
(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./my-cat unavailable

```

```

No files given.
(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./my-cat unavailable
my-cat: cannot open file.
error: No such file or directory

```

my-grep:

For my-grep I copied the functionality of the file reading and writing from my-cat. I used `strstr` to check whether the line from the file contained the given search term and printed the line if the term was present.

Next I worked on the segment of code, which would run if the user gave a search term, but not a file to search from. The code is mostly similar to the previous section, but the checking for no more lines had to be modified a little.

I lastly made sure that the program worked like it was specified in the details and did some error handling.

Program in action (Picture 1: single file given, Picture 2: Multiple files given, Picture 3: no files given, Picture 4: no search term):

```
(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./my-grep this testtext2.txt
Isn't this nice.
(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./my-grep this testtext2.txt testtext4.txt
```

```
(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./my-grep this testtext2.txt testtext4.txt
Isn't this nice.
this is a test of standard input
you should see this line in the output because it has words in it
this line also has words
but this one doesnt
nor does this one
```

```
(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./my-grep this
Empty line or only enter ends the loop.
This is a line that you will not see
But this is a line you will see
But this is a line you will see
```

```
(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./my-grep
my-grep: searchterm [file ...]
(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$
```

my-zip:

The idea of this code was to go through a file first one line at a time and then go through the line one character at a time. If the current char and the next char are the same I added them to a counter, and when the current char didn't match the next char anymore I wrote the number of times the char was present in 4 byte binary code to stdout and after that I wrote the char and returned the counter to 1 for the next char.

After the main part of the code was working I looked through the details given in the assignment and made sure that the code was working in the wanted way and did error handling.

My-unzip:

This program was by far the most difficult and I had a lot of issues with it, although it's working like it's supposed to now. I started by getting the byte size of the zipped file inputted. Then I used an while loop with an int i (which goes by increments of 5) which runs as long as i is smaller than the byte size of the file. Then I used fseek to go to the byte that i is pointing at (0,5,10,15...), this is where the binary numbers are. I used fread to convert the numbers back to int. After this I used fseek again to go to the i+4:th byte, which is where the char is. I used getc to get the char and then converted it to string. After this I made a loop that ran for as long as the read number informed and printed the char each time. Then the loop goes to the next char and so on until the file is fully unzipped. Lastly I checked all the information in the assignment and did all of the error handling.

Examples of my-zip and my-unzip:

(Picture 1: zipping and unzipping a single file, Picture 2: zipping and unzipping multiple files)

textfile1: aaaabbbbccccccddwwq

textfile2: ssssss pppp wwwwwaaaassssdddd

```
(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./my-zip testzip.txt>outputzip.z
(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./my-unzip outputzip.z
aaaabbbbccccccddwwq(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$
```

```
aaaaaaaaaaaaaaaaaaaaaqqq(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$  
(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./my-zip testzip.txt testzip2.txt>outputzip.z  
(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$ ./my-unzip outputzip.z  
aaaabbbbccccccddwwqssssss pppp wwwaaaaassssddddd(base) teodora@teodora-HP-Compaq-Pro-6300-MT:~$
```

Assignment 4:

As getpid() was mentioned in the assignment, I started looking through the files, searching for where the term pid was present. The files that came up were proc.h/c, sysproc.c, syscall.c/h, user.h, usys.S and Makefile. It's easiest to begin from the simplest files so I started with syscall.h, where it is possible to give a system call number to the new system call. I added SYS_getreadcount 22 to the file.

Next I went to the user.h file, where I added "int getreadcount(void)", which will enable user to be able to access this system call. Lastly I went to usys.S and added SYSCALL(getreadcount), now it's possible to create a trap, which will initiate the processing of getreadcount.

The proc files hold information on the beginning of a process, processes are initiated there and for example when a process is found the process id(pid) is created. CountRead is added to the proc struct where pid is also kept, this will enable the tracking of the count by process. Additionally countRead is initiated also when process is found.

Next the system call function is created in sysproc.c. The sysproc.c file contains the functions of some of the system calls, which give the return value of the call. Additionally from this file it is able to access structs in proc files. The idea is that the function will return the count of read that is kept in the proc struct. The implementation is the same as in the getpid function.

Next I went to syscall.c to add the getreadcount function to the list of external functions and to add the return path. Lastly I implemented the counter in systemcall.c. This is done by checking whether the current system call is the same as SYS_read, and if it is the counter is increased by 1. When the current system call is getreadcount then the information in the counter is added to the variable countRead in proc struct of current process.

Next I created the getcountread.c, which prints the output of the system call. I also edited the Makefile a little bit. Mainly to add one text file so that I can test the output of the system call and also I added the getreadcount necessary information so that it will be possible to call it from XV6 terminal. Additionally I added a few lines in grep.c so that when grep is called the getreadcount function will also be called to print the number of times read was called.

I started doing this project a little bit late, so this was all the functionality I could get done in this assignment, but I think it should fulfill the minimum requirements.

System call in action:

```
SeaBIOS (version 1.13.0-1ubuntu1.1)

iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CB00+1FECCB00 CA00

Booting from Hard Disk...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap star
t 50
init: starting sh
$ getreadcount
13
$ cat test.txt
This is a text file
For testing out the cat process
lets see if this works
$ getreadcount
41
$ grep this test.txt
lets see if this works
62$ _
```

References for Assignment 3:

<https://github.com/remzi-arpacidusseau/ostep-projects/blob/master/initial-xv6/background.md>

<https://github.com/MWessel27/xv6-systemcall>

https://medium.com/@flag_seeker/xv6-system-calls-how-it-works-c541408f21ff

<https://gist.github.com/bridgesign/e932115f1d58c7e763e6e443500c6561>

<https://viduniwickramarachchi.medium.com/how-to-add-a-user-program-to-xv6-1209069feee4>

<https://viduniwickramarachchi.medium.com/add-a-new-system-call-in-xv6-5486c2437573>

<https://www.geeksforgeeks.org/xv6-operating-system-add-a-user-program/>

https://pekoeko11.sakura.ne.jp/unix_v6/xv6-book/en/The_first_process.html

<https://github.com/YehudaShapira/xv6-explained/blob/master/Explanations.md>

<https://stackoverflow.com/questions/8021774/how-do-i-add-a-system-call-utility-in-xv6?rq=1>

Repository url (all files):