CT30A3370

Project 1: Warmup to C and Unix programming

Documentation

Joonas Liedes

Requirements

The objective was to implement a program to reverse the contents of a given input file and then write the reversed content in to an output file. Input and output files are given as command line parameters. If no parameters are given, the program defaults to getting input from stdin and outputting it to stdout. Similarly, if only one command line parameter is given, the parameter is considered as the input file. Error handling regarding I/O-operations and memory allocation was required. For detailed descriptions of error handling requirements, please see project description.

Implementation

The development was done in Windows sub system for Linux 2 (WSL2) using an Ubuntu 20.04 distribution. In addition, testing was also conducted in a pure Linux environment using VMware Player running an Ubuntu 20.04 distribution.

The program was implemented as a command line utility that accepts up to 2 arguments in addition to the filename. The first parameter is used for passing the input file and the second parameters is used for determining the output file. A simple if-else structure is implemented in the main method to distinguish between the possible scenarios depending on the number of command line arguments provided. This is done by getting the argument count from the main method. A struct was constructed to represent an individual line in a singly linked list. A utility function newLine() was constructed to simplify creation of new line instances as nodes in a linked list. Malloc was used for dynamic memory allocation. A function for reading the input was implemented with the help of the getline() function available from the standard I/O-library. This function utilizes memory reallocation to ensure there's always enough space. Type of input is checked with strcmp that returns zero if input parameter given to the readFile() function is stdin and numerical value larger or smaller than zero if parameter is not stdin. Contents of the read line are then stored in the newly created node of the linked list with the strdup() function. Finally, memory allocated is then free'ed within the readFile function for the buffer and at the end of the main function via the freeList() function.

The linked list is then reversed with a recursive function called reverseInput(). The function takes a pointer to a pointer as a parameter, that references to the first item in the singly linked list. The list is then divided to the first item and the rest of the items. The rest are then recursively reversed and the first is finally linked to the end.

The reversed input is then written to the output with the function writeOutput(), that takes a pointer to the start of the linked list and the argument vector as parameters. The content of the argument vector is again checked with strcmp() and the output is then directed to a file or stdout depending on the result.

Error handling regarding I/O-operations and memory handling was implemented in the appropriate subroutines responsible for these operations. In addition, errors regarding the usage more directly related to the user were considered in the main() method.

The program was then tested with a sample file resembling the one given in the project description. No errors were found, and the run time was instantaneous. However, when tested with a file containing a million lines (long_input.txt in repository), the program crashed due to a segmentation fault as the recursion depth got too deep. Further testing showed that the program runs with the recursive reversing function up to a file size of ca. 10^5 lines. Subsequently, an iterative approach for reversing the linked list was implemented. With this approach no segmentation faults were discovered. Furthermore, Valgrind was used to check for memory leaks, of which none were found.

Screenshot of Valgrind summary:

```
joonas@DESKTOP-FSABGMJ:/mnt/c/Users/joona/Työt ja opiskelu/LUT/Käyttöjärjestelmäohjelmointi/projektit/GitHub/p1$ valgrind --leak-check=full ./reverse input.txt
 output.txt
==1008== Memcheck, a memory error detector
==1008== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==1008== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==1008== Command: ./reverse input.txt output.txt
==1008==
==1008==
==1008== HEAP SUMMARY:
==1008==     in use at exit: 0 bytes in 0 blocks
==1008==   total heap usage: 13 allocs, 13 frees, 2,177 bytes allocated
==1008==
==1008== All heap blocks were freed -- no leaks are possible
==1008==
==1008== For lists of detected and suppressed errors, rerun with: -s
==1008== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
joonas@DESKTOP-FSABGMJ:/mnt/c/Users/joona/Työt ja opiskelu/LUT/Käyttöjärjestelmäohjelmointi/projektit/GitHub/p1$
```

Screenshot of running program using stdin and stdout:

```
joonas@DESKTOP-FSABGMJ:/mnt/c/Users/joona/Työt ja opiskelu/LUT/Käyttöjärjestelmäohjelmointi/projektit/GitHub/p1$ ./reverse
Enter lines ('q' to quit):
eka
toka
kolmas
q
kolmas

toka

eka

joonas@DESKTOP-FSABGMJ:/mnt/c/Users/joona/Työt ja opiskelu/LUT/Käyttöjärjestelmäohjelmointi/projektit/GitHub/p1$
```