Programming in C II, Group B
Jacobs University Bremen
Dr. Kinga Lipskoch
Xu He

Course: JTSK-320112
February 23rd, 2017

Assignment 6 - Queues and Working with Files

- The problems of this assignment must be solved in C.
- Your programs should have the input and output formatting according to the testcases listed after the problems.
- Your programs should consider the grading rules: http://minds.jacobs-university.de/sites/default/files/uploads/teaching/CProgrammingSpring18/Grading-Criteria-C2.pdf

Problem 6.1 *Counting words in a file* (2 points)

Presence assignment, due by 18:30 h today

Write a program which reads the content of a file given as input and counts the number of the words in the file. It is assumed the words are separated by one or multiple of the following characters: `' ' ','` `'?'` `'!'` `'.'` `'\t'` `'\r'` `'\n'`.

For testing your solution to this problem, please use:

`https://grader.eecs.jacobs-university.de/courses/320112/c/words.txt`

`https://grader.eecs.jacobs-university.de/courses/320112/c/words2.txt`

You can assume that the content of the input file will be valid if existing.

Testcase 6.1: input

```
words.txt
```

Testcase 6.1: output

```
The file contains 17 words.
```

Problem 6.2 *Removing from the queue* (2 points)

Extend the source code of `queue.c` from **Problem 5.4** by implementing the `dequeue()` function. Follow the hints given in the slides (see Lecture 5 & 6, slide 17) and consider the case of a queue underflow.

You can assume that the input will be valid except the semantical possibility of reaching queue underflow. To pass the testcases your output has to be identical with the provided ones.

Testcase 6.2: input

```
a
3
a
5
a
7
d
d
q
```

Testcase 6.2: output

```
add int: Putting 3 into queue
1 items in queue
Type a to add, d to delete, q to quit:
add int: Putting 5 into queue
2 items in queue
Type a to add, d to delete, q to quit:
add int: Putting 7 into queue
3 items in queue
Type a to add, d to delete, q to quit:
Removing 3 from queue
2 items in queue
Type a to add, d to delete, q to quit:
Removing 5 from queue
1 items in queue
Type a to add, d to delete, q to quit:
Bye.
```

Problem 6.3 *Printing the queue* (2 points)

Extend the source code of `queue.h`, `queue.c` and `testqueue.c` from **Problem 6.2** by adding and implementing the additional function `printq()` for printing the elements of the queue separated by spaces. If you enter `'p'`, then the program should print the elements of the queue. Make sure that you can print more than once.
You can assume that the input will be correct. To pass the testcases your output has to be identical with the provided ones.

Testcase 6.3: input

```
a
3
a
5
a
7
p
q
```

Testcase 6.3: output

```
add int: Putting 3 into queue
1 items in queue
Type a to add, d to delete, p to print, q to quit:
add int: Putting 5 into queue
2 items in queue
Type a to add, d to delete, p to print, q to quit:
add int: Putting 7 into queue
3 items in queue
Type a to add, d to delete, p to print, q to quit:
content of the queue: 3 5 7
3 items in queue
Type a to add, d to delete, p to print, q to quit:
Bye.
```

Problem 6.4 *"Authentification" with files* (3 points)

Write a program which reads from the standard input a username and a password, and prints a message on the screen for granting or denying access depending on the fact if the user and the corresponding password are listed in a file which is given as input to the program.
The program should repeatedly check the username and its password introduced from the standard input until the word "`exit`" is introduced for the username. It is assumed that the word "`exit`" is not contained in the input file.
Do not store the whole information (list of usernames and passwords) in the main memory or do not read the whole information from the file for every request, but store only partial information (e.g., the usernames) and use the functions `ftell()` and `fseek()` to read the rest of the needed information.
For testing your solution to this problem, please use:
`https://grader.eecs.jacobs-university.de/courses/320112/c/users.txt`
`https://grader.eecs.jacobs-university.de/courses/320112/c/users2.txt`
You can assume that the content of the input file will be valid if existing.

Testcase 6.4: input

```
users.txt
ben
12b43
kate
jfd45
bill
iu3556
exit
```

Testcase 6.4: output

```
Access to user ben is granted.
Access to user kate is denied.
Access to user bill is granted.
Exiting ...
```

Problem 6.5 *Concat `n` files* (3 points)

Write a program which reads from the standard input the value of an integer `n` and then the names of `n` files. The program should concatenate the content of the `n` files separated by `'\n'` and write the result on the standard output and also into `output.txt`.
Read the input files and write the output file using the binary mode. Use a char buffer of size 64 bytes and chunks of size 1 byte when reading and the same buffer with chunks of size 64 bytes (or less for the last write if file size is not a multiple of 64) when writing.

For testing your solution to this problem, please use:
https://grader.eecs.jacobs-university.de/courses/320112/c/file1.txt
https://grader.eecs.jacobs-university.de/courses/320112/c/file2.txt
https://grader.eecs.jacobs-university.de/courses/320112/c/file3.txt
You can assume that the content of the input files will be valid if existing.

Testcase 6.5: input

```
3
file1.txt
file2.txt
file3.txt
```

Testcase 6.5: output

```
Concating the content of 3 files ...
The result is:
The first file's
content.
The second file's content.
The
third files's
content.
The result was written into output.txt
```

Bonus Problem 6.6 *Binary read and write* (2 points)

Write a program which reads using `fread` the first two characters (separated by space) from the file "`chars.txt`" and writes using `fwrite` the sum of their ASCII code values as an integer into "`codesum.txt`". Use an editor to create the input file "`chars.txt`". Your program is responsible to create the output file "`codesum.txt`".
You can safely assume that the content of the input file will be valid and the two characters are separated by a space.

How to submit your solutions

- Your source code should be properly indented and compile with `gcc` without any warnings (You can use `gcc -Wall -o program program.c`). Insert suitable comments (not on every line . . .) to explain what your program does.
- Please name the programs according to the suggested filenames (they should match the description of the problem) in Grader. Otherwise you might have problems with the inclusion of header files. Each program **must** include a comment on the top like the following:

```
/*
   JTSK-320112
   a6_p1.c
   Firstname Lastname
   myemail@jacobs-university.de
*/
```

- You have to submit your solutions via *Grader* at
  `https://grader.eecs.jacobs-university.de`.
  If there are problems (but **only** then) you can submit the programs by sending mail to `x.he@jacobs-university.de` **with a subject line that begins with JTSK-320112**.
  It is important that you do begin your subject with the coursenumber, otherwise I might have problems to identify your submission.
- Please note, that after the deadline it will not be possible to submit any solutions. It is useless to send late solutions by mail, because they will not be accepted.

This assignment is due by Wednesday, February 28th, 10:00 h.