

Homework: C File Processing

This document defines the homework assignments from [the "C Programming" Course @ Software University](#). Please submit as homework a single **zip / rar / 7z** archive holding the solutions (source code) of all below described problems.

Problem 1. Print File Contents

Write a program that reads a text file and prints its contents on the console.

Problem 2. Odd Lines

Write a program that reads a text file and prints on the console its odd lines.

Problem 3. Line Numbers

Write a program that reads a text file and inserts line numbers in front of each of its lines. The result should be written to another text file.

program.c	modified.c
<pre>#include <stdio.h> int main() { printf("Spicy"); return 0; }</pre>	<pre>0 #include <stdio.h> 1 2 int main() 3 { 4 printf("Spicy"); 5 6 return 0; 7 }</pre>







Problem 4. Copy Binary File

Write a program that copies the contents of a binary file (e.g. image, video, etc.) to another file.







Problem 5. Slicing File

Write a program that takes any file and slices it to **n** parts. Write the following functions:

- slice(const char *sourceFile, const char *destinationFile, size_t parts)** - slices the given source file into **n** parts and saves them in **destinationDirectory**.

Source File	Destination Directory
<p>parts = 5</p> <div> SOLID-Logger.avi 00:32:06 678 MB</div>	<div><div> Part-0.avi 135 MB</div><div> Part-2.avi 135 MB</div><div> Part-4.avi 135 MB</div></div> <div><div> Part-1.avi 135 MB</div><div> Part-3.avi 135 MB</div></div>

- **assemble(const char **parts, const char *destinationDirectory)** - combines all parts into one, in the order they are passed, and saves the result in **destinationDirectory**.

Source Files		Destination Directory
 Part-0.avi 135 MB	 Part-1.avi 135 MB	 assembled.avi 00:32:06 678 MB
 Part-2.avi 135 MB	 Part-3.avi 135 MB	
 Part-4.avi 135 MB		

The **input file names**, **destination directory** and **parts** should be passed to the program as arguments.

The program should produce proper error messages in case of errors. Use buffered reading.

Problem 6. Fix Subtitles

Write a program that takes as arguments **input subtitles file** and **offset in milliseconds**. The program should edit the subtitles' timing by the given offset. The subtitles will contains will be in the format specified below.




```
#num
hh:mm:ss,ms --> hh:mm:ss,ms
text
```

The program should correctly modify seconds, minutes and hours when overflow occurs (i.e. 61 seconds is not valid). Example: **00:00:52,580 + 700 -> 00:00:53,280**. The program should support modifying subtitles in the range [00:00:00,000 - 99:59:59,999].

Offset	source.sub	fixed.sub
1500	44 00:04:22,535 --> 00:04:24,870 (Laughs) Take her.	44 00:04:24,035 --> 00:04:26,370 (Laughs) Take her.
	45 00:04:24,904 --> 00:04:28,874 Are you men or snakes, that you would threaten a child?	45 00:04:26,404 --> 00:04:30,374 Are you men or snakes, that you would threaten a child?

Problem 7. * Directory Traversal

Traverse a given directory for all files with the given extension. Search through the first level of the directory only and write information about each found file in **report.txt**.

Input	Directory View	report.txt
/home/pesho/Downloads	 C-Ref-Files  1. C-Programming-Intro.pdf  2. C-	3. C-Programming-Formatted-IO.pdf - 3243KB 2. C-Programming-Data-Types.pdf - 6834KB C-Ref-Files - 4KB cpp.chm - 472KB 10. C-Programming-Memory-Management-Exercises.zip - 34KB refs.chm - 468KB 1. C-Programming-Intro.pdf - 2525KB netbeans-8.0.2-cpp-linux.sh - 64918KB

Problem 8. ** Full Directory Traversal

Modify your previous program to **recursively traverse** the **sub-directories** of the starting directory as well.

Problem 9. *** Word Count

Write a program that reads a list of words from the file **words.txt** and finds how many times each of the words is contained in another file **text.txt**. Matching should be **case-insensitive**.

Write the results in file **results.txt**. Sort the words by frequency in descending order.

words.txt	text.txt	result.txt
quick is fault	-I was quick to judge him, but it wasn't his fault. -Is this some kind of joke?! Is it? -Quick, hide here...It is safer.	is - 3 quick - 2 fault - 1