HW 2 Notes

First, unbuffer standard output in main with the following code:

setvbuf( stdout, NULL, \_IONBF, 0 );

For each command line argument (call pipe() for communication between parent and child process and then call fork() to generate a child process for each file 🡪 in each child process (1 process per file for n files so n total processes), open and read each file and extract valid words and write them to a pipe which will connect to a separate executable running on Submitty by writing valid words and append after each word with . to pipe

Parent will call fork() and child process will call executable with execl

Parent process must call waitpid() on each child process (this will synchronize all child processes working on parsing the files along with the other child process calling the executable)

Do not use square brackets anywhere in the code or comments – they will be removed!

To be counted as a word, it must have at least 2 or more alphanumeric characters (use isalnum()) and assume that no word is more than 127 bytes long, all other characters are delimiters, and words are case sensitive 🡪 you can use same logic as in homework 1 for parsing the file for valid words

Pipe between parent and child: Parent reads, child writes

Pipe between child and executable: Child writes, executable reads (parent will create another separate child process via fork() which will then call executable “hw2-cache.out” with execl and pass pipe read descriptor as command line argument to this executable)

Output:

When each child process returns, it must return one of the following values:

3 if no valid words found in the input file

2 if the given input file is not found

EXIT\_FAILURE (1) if some other error occurred

EXIT\_SUCCESS (0) if everything worked as expected

Sample output:

PARENT: Created pipe successfully

PARENT: Calling fork() to create child process to execute hw2-cache.out...

PARENT: Calling fork() to create child process for "lion.txt" file...

CHILD: Successfully wrote XXX words on the pipe

PARENT: Child process terminated with exit status 0

PARENT: Child running hw2-cache.out terminated with exit status 0

One of two possible outputs for child process with abnormal termination:

PARENT: Child process terminated abnormally

OR

PARENT: Child running hw2-cache.out terminated abnormally

Miscellaneous error handling:

If improper command-line arguments are given, report an error message to stderr and abort further program execution. In general, if an error is encountered, display a meaningful error message on stderr by using either perror() or fprintf(), then aborting further program execution. Only use perror() if the given library or system call sets the global errno variable.

Error messages must be one line only and use the following format:

ERROR: <error-text-here>