Name: Evan McCann

Design of wc_mul.c:

Mult-process Structure:

wc_mul.c uses multi-process and the parent process divides the input file into equal chunks for each child process. The number of children are based on the amount given in the input. It assigns each chunk by getting the total size (fseek/ftell) and then dividing it by number of children.

IPC:

child:

Each child has its own pipe and is forked. It closes the read end and then opens the file and calls word_count with its given offset and size. It then writes count_t to write end of the pipe and then exits. The last child gets all the remainder bytes, like if there is an odd number so the whole file goes through the code properly.

parent:

The parent closes all write ends after forking the children, opposite of children. It calls waitpid for each child, checks exit status, reads count_t and then takes all the line, word, and character counts and prints the final total

Handling crashes:

Successful children:

Parent waits with waitpid and if children successful their results are sent to parent.

Failed children:

I used an internal loop in the parent where, if a child fails, it is immediately retried.

I did this because, if a child fails, retrying immediately versus waiting did not seem like a big deal.

Also immediate seemed much easier to code.

I thought about it similar to RTT and packets going missing from Computer Networks.

I figured each child was its own packet and if one fails I need to go back as soon as possible so why not do it immediately.

Testing performance:

When testing performance was fairly similar on a macro scale. For the large.txt file 10 processes had a better real time than 1 process.

For the small.txt 1 thread was better as it was a small enough file to not need more processes and adding more ended up only adding to time.

When crashes were added, the more processes the better for large.txt.

However the opposite was true for small.txt. 1 process was always better, though only slightly compared to 10, but the difference was much greater at 20.

Example of both with the given commands in the PPT:

```
ecm19269@csci-odin:~/Operating Systems/PA 1/Project_Code/001.release$ time ./wc_mul 1
large.txt 0
CRASH RATE: 0
[pid 3781440] reading 101763265 bytes from offset 0

========== Final Results ================
Total Lines : 2000008
Total Words : 19082237
Total Characters : 82681028
==========================================

real    0m0.689s
user    0m0.632s
sys     0m0.054s
ecm19269@csci-odin:~/Operating Systems/PA 1/Project_Code/001.release$ time ./wc_mul 10
large.txt 0
CRASH RATE: 0
[pid 3781445] reading 10176326 bytes from offset 0
[pid 3781446] reading 10176326 bytes from offset 10176326
[pid 3781447] reading 10176326 bytes from offset 20352652
[pid 3781448] reading 10176326 bytes from offset 30528978
[pid 3781449] reading 10176326 bytes from offset 40705304
[pid 3781450] reading 10176326 bytes from offset 50881630
[pid 3781451] reading 10176326 bytes from offset 61057956
[pid 3781452] reading 10176326 bytes from offset 71234282
[pid 3781453] reading 10176326 bytes from offset 81410608
[pid 3781454] reading 10176331 bytes from offset 91586934

========== Final Results ================
Total Lines : 2000008
Total Words : 19082237
Total Characters : 82681028
==========================================

real    0m0.080s
user    0m0.673s
sys     0m0.069s
```