

---

## COS 350: Program #6: multiple file line counting

**Objectives:** Working with fork, exec, pipes and comparing with pthreads and shared memory communication.

---

You will write 3 versions of a multiple file line counting program. Your goal is to make all three versions as fast as possible. Do not use `getc()` as in the class examples. Use `read()` to read in a large chunk of the file into a buffer for processing. This will be faster.

### Version 1: lc1

Works like `wc` (word count) but simply counts the number of lines in each file specified on the command line, and for each file it prints the count and the filename, and finally the total. (Just count the number of newline characters. Your results should match `wc`.)

### Version 2: lc2

Use `fork()` and `exec()` to create a separate process for each individual file. Just `exec()` `lc1` to do the counting and setup pipes to read back the output so that the main program can total them up. The output should be identical to `lc1` except the output order may vary. Fork them all off so that they are working simultaneously.

### Version 3: lc3

Use `pthread`s to count the files. Create simultaneous threads, one for each file. You can decide what sort of synchronization you need. As with `lc2`, the output order may vary. Remember to use the `-pthread` compiler flag.

### Testing:

Once you have your programs all working, recompile them using the optimization flag `-O2` (e.g. `gcc -O2 ...`). This will make them run faster.

Use the unix program *time* for timing your programs. Time each of these running on the set of 10 files: `f1 – f10` (these are in the course directory). Do the timing tests on an unloaded machine. You can check the load on a machine with the program: `uptime`. The load values should be  $< 1.0$ . Preferably close to 0.0.

Run at least 3 trials for each version and report the best time among these trials. Which version is fastest? Try to guess why.

**What to turn in:**

For each assignment you will turn in a written part in class as well as submitting your code electronically.

**Written part:**

- Your name.
- A discussion of any incomplete parts, known bugs, deviations from the specification, or extra features in your program.
- A printout of your code. Use a2ps to save paper.
- A printout of your Makefile
- A log of the following testing results. (I recommend using script to capture it.)

<code>time lc1 f*</code>	<i>test lc1 counting all 10 files f1-f10</i>
<code>time lc2 f*</code>	<i>test lc2</i>
<code>time lc3 f*</code>	<i>test lc3</i>
<code>exit</code>	<i>exit script</i>

- The filled in testing sheet with your timing results and explanations.

**Electronic submission:**

- Before you submit your program, compile and test it on a **linux** machine in the lab.
- From a machine in the lab run the program “submit” to submit your files.
- Submit your source code (**lc1.c lc2.c lc3.c**) and any other source files, a **Makefile** that works for all 3 files, and your executables (**lc1 lc2 lc3**) to a directory named: **prog6**
- **Do not in any way combine, compress, zip, or tar your files!**

**Grading:**

See testing sheet.

COS 350 Program 6 Testing - please fill out and turn in this testing sheet with your written submission.

Name(s): \_\_\_\_\_

	Pts	Works	Has problems (explain)	Not implemented	Explanation
lc1 gets correct individual file line counts	15				
lc1 gets correct total	5				
lc2 the processes execute concurrently (you should see a speedup over lc1)	15				
lc2 gets correct total	15				
lc3 the threads execute concurrently (you should see a speedup over lc1)	15				
lc3 gets correct total	15				

Timing results (10 points):

Machine that the timing tests were performed on: \_\_\_\_\_

Try to use an unloaded machine and report the best of at least 3 trials for each program.

Timings results	real	user	sys
lc1			
lc2			
lc3			

(10 points) Which version was the fastest, and why do you think that was?