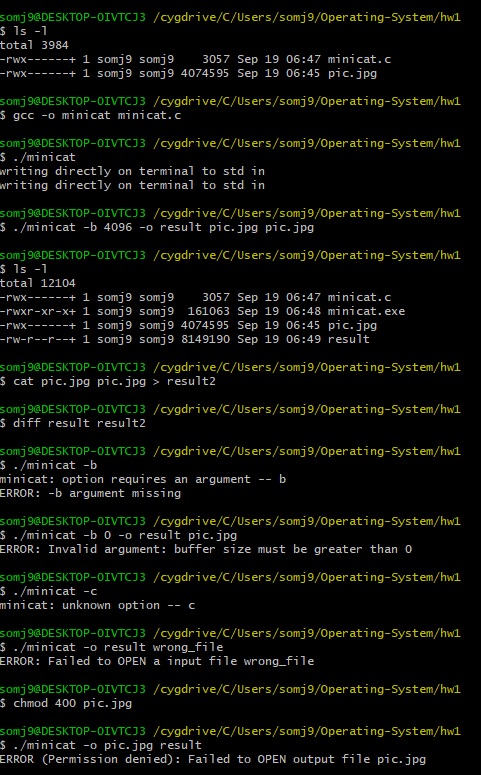
ECE 357 Computer OS

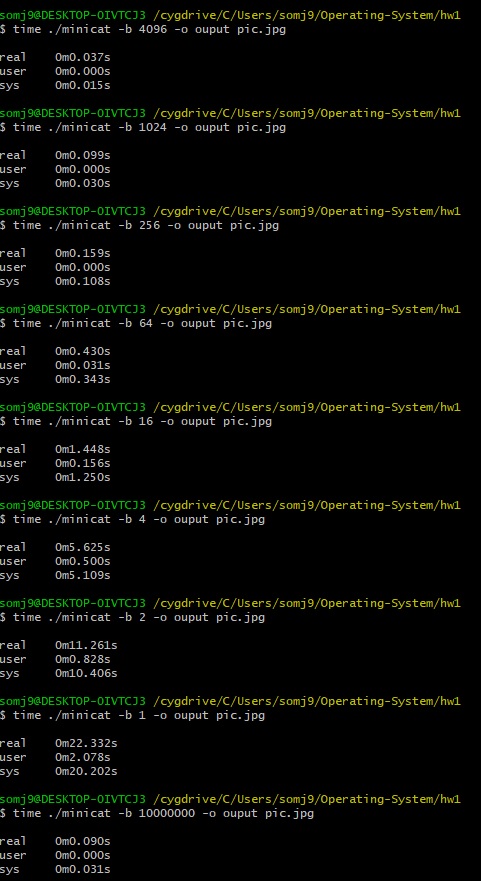
September 19, 2018

Min Joon So

**Screenshot of successful run of my program and error detection/report**

****

**Experimental raw data**

****

**Analysis of results**

As it can be seen from the result, the performance increases (or, the time it takes to run the program decreases) as buffer size increases. Buffer size and performance almost seem to have linear relationship if you compare performance vs buffer size when buffer is set low (1~64). After some point however, increasing buffer doesn’t necessarily increase the performance, but rather slows down the program. I allocated a really big amount of memory (100,000) for buffer to test and it actually slowed downed the program, when compared to allocating 4096 for buffer. The optimum buffer size seems to be around this number. It is possible that the optimal buffer size can changed based on the size of the file I am concatenating, so it would be wise to change the buffer size based on input files.

**Question to ponder:** *How can you specify an input file which is literally a single hyphen, and not have it be confused with a command-line flag or the special symbol for standard input?*

One way is to code in another option flag for input files. For example, in addition to *-o* and *-b* flag, we can add *-i* flag to explicitly tell the program that the next argument is the name of a file, not the special symbol for standard input:

minicat [-b ###] [-o outfile] [-i -] infile1 infile2

minicat -b 1024 -o output -i - infile1 infile2

This concatenates ‘-‘, ‘infile1’ and ‘infile2’ where as

Minicat -b 1024 -o ouput - infile1 infile2

concatenates std input, ‘infile1’ and ‘infile2’

**Program source code listing**