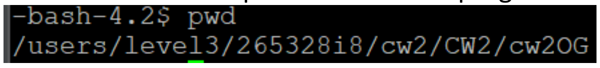
Systems Programming Assessed Exercise 2a: Concurrent Dependency Discoverer

Authorship statement
This is my own work as defined in the Academic Ethics agreement I have signed.

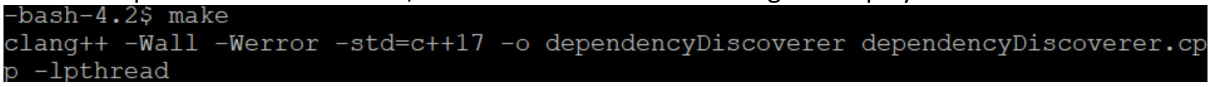
Status
The multithreaded solution I have provided compiles without errors or warnings and appears to be working as intended in all the testing I have done shown below.

Sequential & 1-Thread runtimes
Shown below is the path at which the program is executing.



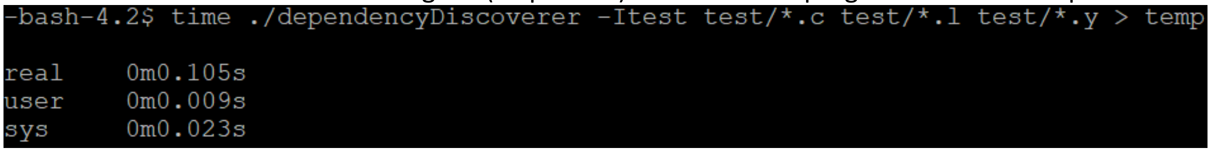
```
-bash-4.2$ pwd
/users/level3/265328i8/cw2/CW2/cw2OG
```

When compiled with the Makefile, there are no errors or warnings as displayed here



```
-bash-4.2$ make
clang++ -Wall -Werror -std=c++17 -o dependencyDiscoverer dependencyDiscoverer.cp
p -lpthread
```

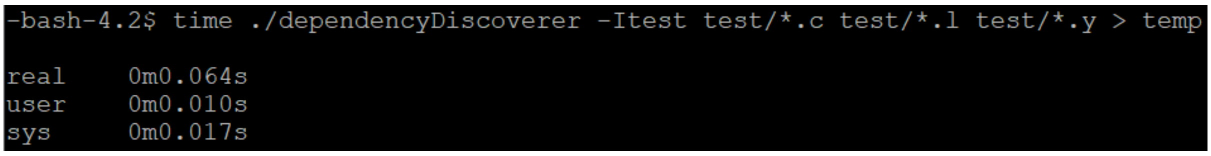
This is the time taken to run the original (sequential) version of the program at the above path



```
-bash-4.2$ time ./dependencyDiscoverer -Itest test/*.c test/*.l test/*.y > temp

real    0m0.105s
user    0m0.009s
sys     0m0.023s
```

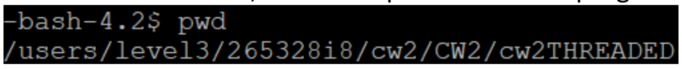
Now running in a new path so as to not interfere with the original version, this is running the threaded version with just one thread. Its path is /users/level3/265328i8/cw2/CW2/cw2THREADED/ as will be shown in a screenshot in a future section.



```
-bash-4.2$ time ./dependencyDiscoverer -Itest test/*.c test/*.l test/*.y > temp

real    0m0.064s
user    0m0.010s
sys     0m0.017s
```

Runtime with Multiple Threads.
As mentioned earlier, this is the path where the program is run to obtain non-sequential runtimes.



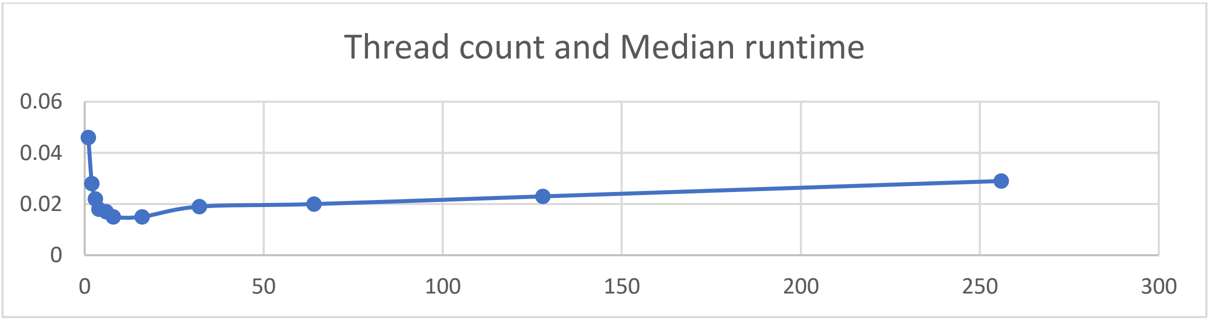
```
-bash-4.2$ pwd
/users/level3/265328i8/cw2/CW2/cw2THREADED
```

To show the benefit of running multiple threads, this table shows the elapsed real times running when specifying various thread counts.

CRAWLER_ THREADS	1	2	3	4	6	8
	Elapsed time(s)	Elapsed time(s)	Elapsed time(s)	Elapsed time(s)	Elapsed time(s)	Elapsed time(s)
Execution 1	0.050	0.029	0.022	0.020	0.019	0.015
Execution 2	0.050	0.028	0.024	0.021	0.019	0.015
Execution 3	0.048	0.027	0.023	0.020	0.019	0.015
Median	0.050	0.028	0.023	0.020	0.019	0.015

This table clearly shows that running with 1 thread is substantially slower than 8, however this theme does not continue when increasing the thread count even further as more time will be spent creating and switching between them than will be spent executing the code, resulting in a longer runtime.

CRAWLER_ THREADS	8	16	32	64	128	256
	Elapsed time(s)	Elapsed time(s)	Elapsed time(s)	Elapsed time(s)	Elapsed time(s)	Elapsed time(s)
Execution 1	0.015	0.015	0.018	0.020	0.023	0.028
Execution 2	0.015	0.016	0.019	0.020	0.023	0.029
Execution 3	0.015	0.016	0.020	0.020	0.024	0.029
Median	0.015	0.016	0.019	0.020	0.023	0.029



The number of cores available may have also had an impact on these results. The machine used to obtain these times had 4 cores, but if there were more available cores to run threads concurrently, the runtimes would in turn decrease and the number of threads used before the elapsed time increased would increase as the system could handle more threads.