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

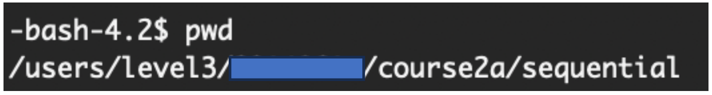
Status

The program is multithreaded solution, and compiles with no error or warning when using Makefile `clang++ -Wall -Werror -std=c++17 -o dependencyDiscoverer dependencyDiscoverer.cpp -lpthread`. When running `../dependencyDiscoverer *.y *.l *.c | diff – output` it shows no output which is correct.

Build, and sequential (i.e., original) & 1-thread runtimes

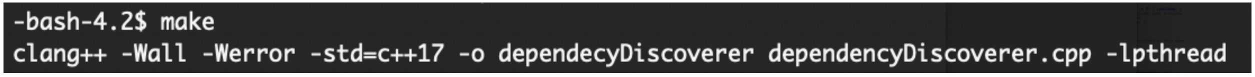
original

a)



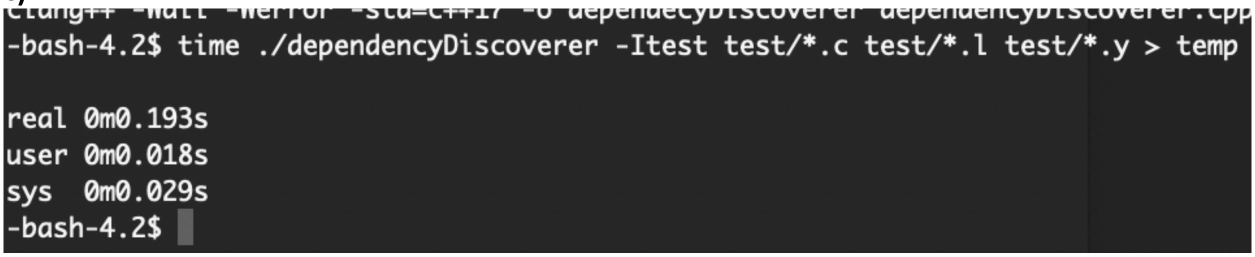
```
-bash-4.2$ pwd
/users/level3/[redacted]/course2a/sequential
```

b)



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

c)

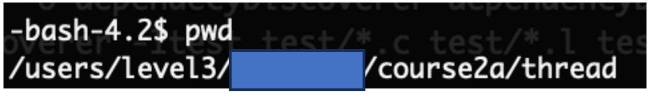


```
clang++ -Wall -Werror -std=c++17 -o dependecyDiscoverer dependencyDiscoverer.cpp
-bash-4.2$ time ./dependencyDiscoverer -Itest test/*.c test/*.l test/*.y > temp

real 0m0.193s
user 0m0.018s
sys  0m0.029s
-bash-4.2$
```

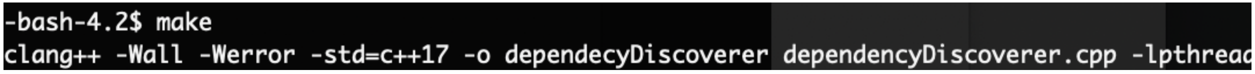
1 Thread

a)



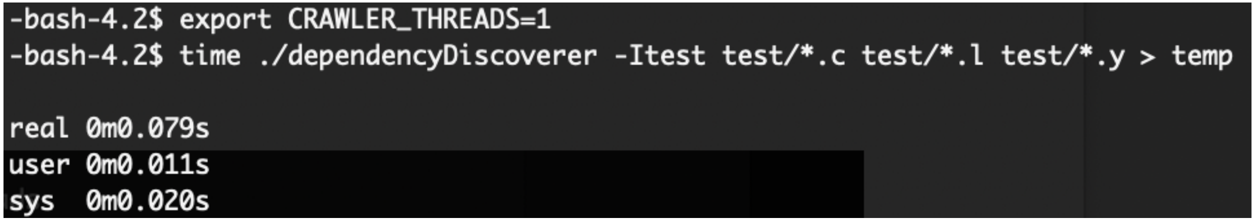
```
-bash-4.2$ pwd
/users/level3/[redacted]/course2a/thread
```

b)



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

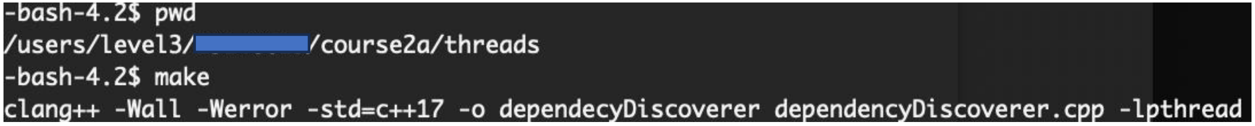
c&d)



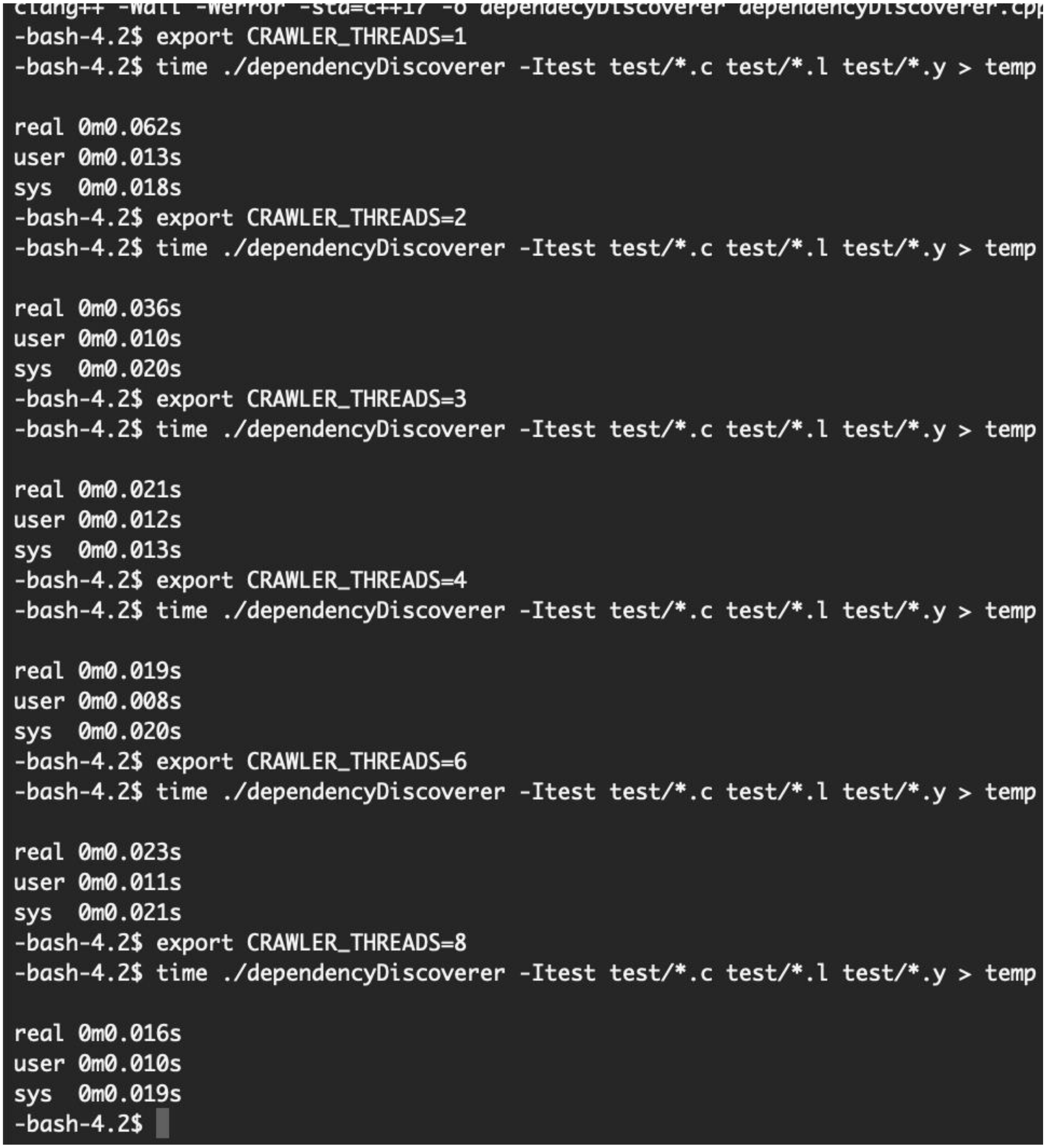
```
-bash-4.2$ export CRAWLER_THREADS=1
-bash-4.2$ time ./dependencyDiscoverer -Itest test/*.c test/*.l test/*.y > temp

real 0m0.079s
user 0m0.011s
sys  0m0.020s
```

Runtime with Multiple Threads



```
-bash-4.2$ pwd
/users/level3/[redacted]/course2a/threads
-bash-4.2$ make
clang++ -Wall -Werror -std=c++17 -o dependecyDiscoverer dependencyDiscoverer.cpp -lpthread
```



```
clang++ -Wall -Werror -std=c++17 -o dependecyDiscoverer dependencyDiscoverer.cpp
-bash-4.2$ export CRAWLER_THREADS=1
-bash-4.2$ time ./dependencyDiscoverer -Itest test/*.c test/*.l test/*.y > temp

real 0m0.062s
user 0m0.013s
sys  0m0.018s
-bash-4.2$ export CRAWLER_THREADS=2
-bash-4.2$ time ./dependencyDiscoverer -Itest test/*.c test/*.l test/*.y > temp

real 0m0.036s
user 0m0.010s
sys  0m0.020s
-bash-4.2$ export CRAWLER_THREADS=3
-bash-4.2$ time ./dependencyDiscoverer -Itest test/*.c test/*.l test/*.y > temp

real 0m0.021s
user 0m0.012s
sys  0m0.013s
-bash-4.2$ export CRAWLER_THREADS=4
-bash-4.2$ time ./dependencyDiscoverer -Itest test/*.c test/*.l test/*.y > temp

real 0m0.019s
user 0m0.008s
sys  0m0.020s
-bash-4.2$ export CRAWLER_THREADS=6
-bash-4.2$ time ./dependencyDiscoverer -Itest test/*.c test/*.l test/*.y > temp

real 0m0.023s
user 0m0.011s
sys  0m0.021s
-bash-4.2$ export CRAWLER_THREADS=8
-bash-4.2$ time ./dependencyDiscoverer -Itest test/*.c test/*.l test/*.y > temp

real 0m0.016s
user 0m0.010s
sys  0m0.019s
-bash-4.2$
```

Experiment

CRAWLER_ THREADS	1	2	3	4	6	8
	Elapsed Time	Elapsed Time	Elapsed Time	Elapsed Time	Elapsed Time	Elapsed Time
Execution 1	0.062	0.036	0.023	0.021	0.024	0.014
Execution 2	0.058	0.031	0.024	0.019	0.018	0.019
Execution 3	0.038	0.029	0.024	0.019	0.019	0.018
Median	0.058	0.031	0.024	0.019	0.019	0.018

Discussion

a) **the benefits of additional cores for this input data**
When we add more cores at first, going from 1 to 2, and 3 the median time it takes to process data drops a lot (from 0.058 -> 0.031 -> 0.024 seconds). This happens because adding more cores allows the system to handle more tasks simultaneously. But when we keep adding more cores after that, like going from 4 to 6, the time saved is very small (from 0.019 to 0.018 seconds). The reason for this is that there's a limit to how effectively the tasks can be divided among cores. Once you reach a certain number of cores, adding more doesn't help as much because the system starts running into other bottlenecks, like the limits of data transfer or the efficiency of the software in splitting tasks.

b) **How Elapsed Times Change**
With a smaller number of cores, each additional core significantly improves processing speed because it allows for more parallel processing. But as the number of cores increases, the benefits start to level off. This shows that while increasing cores is effective up to a point, beyond that, other factors prevent the system from speeding up in the same way.