Systems programming assessed exercise 2

Contents

1. Status:

As far as I am aware, my solution works without errors or warnings when using the command provided in the brief. I have managed to provide a multithreaded solution.

2. Build and sequential (original) & 1 thread runtimes:

a. Path:

I ran the original file on the stlinux02 servers.

```
-bash-4.2$ pwd
/users/level3/255431k9/cw2expectedout
```

b. Crawler compilation (make):

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

c. Time to run sequential:

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

real    0m0.083s
user    0m0.011s
sys     0m0.025s
```

d. Time to run 1 thread:

My single threaded implementation was saved before going on to the multithreaded part so isn't as refined as my final version. I chose to test my single threaded file on VSCode for simplicity, rather than uploading to the server as the brief does not say this part needs to be done on the server.

The pwd for your reference:

```
aliki@AlikisLaptop:/mnt/c/sp/cw2$ pwd
/mnt/c/sp/cw2
```

```
aliki@AlikisLaptop:/mnt/c/sp/cw2$ time ./dependencyDiscoverer -Itest test/*.c test/*.l test/*.y > temp

real    0m0.531s
user    0m0.010s
sys     0m0.133s
```

The result of running it with one thread on the multithreaded version for comparison as in the mark scheme

```
aliki@AlikisLaptop:/mnt/c/sp/cw2$ export CRAWLER_THREADS=1
aliki@AlikisLaptop:/mnt/c/sp/cw2$ time ./dependencyDiscover
er -Itest test/*.c test/*.l test/*.y > temp

real    0m0.632s
user    0m0.000s
sys     0m0.112s
```

3.Runtime with multiple threads

a. Path:

```
-bash-4.2$ pwd
/users/level3/255431k9/cw2
```

Times to run crawler with various thread numbers:

CRAWLER_THREADS= 1:

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

real    0m0.084s
user    0m0.016s
sys     0m0.023s
-bash-4.2$
```

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

real    0m0.081s
user    0m0.010s
sys     0m0.022s
```

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

real    0m0.049s
user    0m0.008s
sys     0m0.017s
```

CRAWLER_THREADS =2

```
-bash-4.2$ export CRAWLER_THREADS=2
-bash-4.2$ time ./dependencyDiscoverer -Itest test/*.c test/*.l test/*.y > temp

real    0m0.039s
user    0m0.008s
sys     0m0.024s
-bash-4.2$
```

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

real    0m0.031s
user    0m0.008s
sys     0m0.020s
```

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

real    0m0.030s
user    0m0.012s
sys     0m0.017s
```

CRAWLER_THREADS =3

```
-bash-4.2$ export CRAWLER_THREADS=3
-bash-4.2$ time ./dependencyDiscoverer -Itest test/*.c test/*.l test/*.y > temp

real    0m0.033s
user    0m0.008s
sys     0m0.026s
-bash-4.2$
```

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

real    0m0.023s
user    0m0.014s
sys     0m0.015s
```

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

real    0m0.025s
user    0m0.009s
sys     0m0.020s
```

CRAWLER_THREADS=4

```
-bash-4.2$ export CRAWLER_THREADS=4
-bash-4.2$ time ./dependencyDiscoverer -Itest test/*.c test/*.l test/*.y > temp

real    0m0.022s
user    0m0.014s
sys     0m0.019s
```

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

real    0m0.029s
user    0m0.009s
sys     0m0.027s
```

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

real    0m0.020s
user    0m0.006s
sys     0m0.023s
```

CRAWLER_THREADS =6

```
-bash-4.2$ export CRAWLER_THREADS=6
-bash-4.2$ time ./dependencyDiscoverer -Itest test/*.c test/*.l test/*.y > temp

real    0m0.030s
user    0m0.014s
sys     0m0.034s
```

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

real    0m0.019s
user    0m0.013s
sys     0m0.017s
```

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

real    0m0.025s
user    0m0.010s
sys     0m0.026s
```

CRAWLER_THREADS =8

```
-bash-4.2$ export CRAWLER_THREADS=8
-bash-4.2$ time ./dependencyDiscoverer -Itest test/*.c test/*.l test/*.y > temp

real    0m0.018s
user    0m0.012s
sys     0m0.023s
```

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

real    0m0.019s
user    0m0.013s
sys     0m0.022s
```

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

real    0m0.023s
user    0m0.012s
sys     0m0.028s
```

b. Experiment

Crawler_thread	1	2	3	4	6	8
Execution 1 (s)	0.084	0.039	0.033	0.022	0.030	0.018
Execution 2 (s)	0.081	0.031	0.023	0.029	0.019	0.019
Execution 3 (s)	0.049	0.030	0.025	0.020	0.025	0.023
Median	0.081	0.031	0.025	0.022	0.025	0.019

c. Discussion:

Having just one thread is very slow. Using additional cores seems to improve the elapsed times however between 2 and 8 threads the elapsed times are between 0.019-0.025 so the additional cores do not add that much variation. Furthermore, 6 worker threads seem to take longer than when there are just 4, which would suggest that adding too many cores results in a longer runtime. However, we see this not the case as 8 worker threads has the smallest median elapsed time for this iteration of the experiment. In general, the larger the number of worker threads, the more efficient and therefore the faster the program is.