Operating Systems
Constructor University
Dr. Jürgen Schönwälder

Module: CO-562
Date: 2026-01-08
Due: 2026-01-15

Problem Sheet #13

This sheet is only for students who failed to obtain the module achievement.

Problem 13.1: *multi-threaded partition of integers into sums of distinct primes* (10 points)

Write a multi-threaded C program called `partition`, which partitions natural numbers concurrently into sums of distinct prime numbers. The program reads the numbers to be partitioned and the count of prime numbers from the standard input and writes the result to the standard output. For example, the input 18 2 requests to partition the number 18 into a sum of 2 prime numbers. Use the lowest primes possible, that is pick 18 = 5 + 13 and not 18 = 7 + 11 or 18 = 13 + 5. Multiple numbers can be partitioned concurrently using the thread pool pattern. The `-t` command line argument controls how many worker threads are created, i.e., the size of the thread pool. A sample invocation is shown below (the shell command uses a here document, which is a block of text passed by the shell to the program as its input):

```
$ ./partition -t 6 <<EOD
a 99809 1
b 18 2
c 19 3
d 20 4
e 22699 1
f 22699 2
g 22699 3
h 22699 4
i 40355 3
EOD
b 18 partitioned with 2 distinct primes: 5 + 13
d 20 cannot be partitioned with 4 distinct primes
c 19 partitioned with 3 distinct primes: 3 + 5 + 11
f 22699 partitioned with 2 distinct primes: 2 + 22697
a 99809 partitioned with 1 distinct primes: 99809
h 22699 partitioned with 4 distinct primes: 2 + 3 + 43 + 22651
e 22699 partitioned with 1 distinct primes: 22699
g 22699 partitioned with 3 distinct primes: 3 + 5 + 22691
i 40355 partitioned with 3 distinct primes: 3 + 139 + 40213
```

Note that every number and counter pair is prefixed with a tag, which can be used to match results to requests. The content of the tag is arbitrary but it is safe to assume that it does not contain white space or control characters. Detailed requirements:

a) The program has to partition numbers in the unsigned 64-bit integer number space correctly.

b) The program must use the thread pool pattern where a fixed number of threads are created to perform the partitioning.

c) The tags and numbers are read from the standard input by an input reading thread and placed into a queue where they are picked up by worker threads.

d) Partitioning results are passed via another queue to an output writing thread.

e) The program must detect the end of the input and shutdown the worker threads.

f) The main thread waits until all worker threads, the reader thread, and the writer thread have terminated.

g) The program must follow the programming requirements stated on the course web page. In particular, it needs to handle runtime error situations in an appropriate manner.