CSE 344 – SYSTEM PROGRAMMING

by

ERCHAN APTOULA

MIDTERM PROJECT REPORT

GÖKHAN HAS - 161044067

May 12, 2020

CSE 344 - SYSTEM PROGRAMMING MIDTERM PROJECT REPORT

In Midterm project, our aim is to provide communication between N + M + 1 processes sent as parameters. I created these communications using semaphores.I defined these semaphores (some of them used as mutex) in main and it was enabled to communicate with other processes using the mmap(.., .. MAP_SHARED, .., ..) function.

Synchronization Between the Supplier and the Cook

Cook should not be there when the Supplier enters the kitchen. If the two are in the kitchen together, there is a deadlock problem. That's why a mutex named mutexKitchen is used. This mutex prevents anyone else from entering if someone is in the kitchen. Likewise, there is a semaphore in the supplier process that keeps the number of free spaces in the kitchen. This semaphore is sizeOfKitchen semaphore. The initial value is up to 2 * L * M + 1. Each time the supplier puts a plate in the kitchen, this semaphore value is reduced by using the sem_wait() function. Thus, once the size of the kitchen is full, the supplier cannot put plates until empty. This may also appear as a zigzag at the top of the latest graphics. The Supplier function takes a file descriptor as a parameter. Returns this file by reading a single byte. Supplier takes 'p', 'c', 'd' or the capital letters of these rotating bytes and brings plates to the kitchen.

Between the cook process (or s) and the supplier process, 3 different semaphore partners are used, keeping the number of food in the kitchen. The supplier increases their value in the kitchen, while the cook process(or s) decreases their value.

Synchronization Between the Cook and the Student

The Cook function is the function that takes the most parameters in the project. This is because Cook's relationship with both the Supplier and the Student. It acts as a bridge between the two. Between cook and student processes there is a mutex named mutexCounter. It should not enter the counter at the same time and cause deadlock. And 3 foods on the counter are kept in 3 different semaphores. This is common in the semaphore. Because the value of these semaphore should increase when Cook puts each plate on the counter. Cook uses an auxiliary function when putting it on the counter. This function returns the food, whichever is less, by looking at the value of the semaphores of 3 different foods on the counter. Therefore, the required food is left to the counter before the counter limit is reached.

Thus, deadlock problem is solved. Otherwise, when the limit of the counter was full, if there was not at least one of the 3 different foods, no student would be able to eat.

Student processes are divided into two different functions. One of them is theGraduated () and the other is theStudent (). Communication between these functions is also provided with mutex. Because graduated students should have a priority. If the graduated student is at the counter, he / she should get food before the normal student.

Again, between these two processes, the emptyTable semaphore is shared. The initial value of this semaphore is equal to the T parameter. When the student is seated, with sem_wait, this decreases the value of semaphore flour, after eating, it increases its value with sem_post.

MAIN FUNCTIONS :

int theSupplier(sem_t* sem_P, sem_t* sem_C, sem_t* sem_D, sem_t* sizeOfKitchen, sem_t* mutexKitchen, int size, int fd): The Supplier function takes many parameters. It takes the plate one by one to the kitchen by reading from the file given with the fd parameter it takes.

sem_P --> number of soups in the kitchen
sem_C --> number of main courses in the kitchen
sem_D --> number of deserts in the kitchen
sizeOfKitchen --> The number of free spaces in the kitchen is initially 2 * L M + 1.
mutexKitchen --> It will be used to prevent both supplier and cook from entering the kitchen at the same time.
size --> (L*M)
fd --> The file descriptor is required for the file that the supplier will read.

int theCook(sem_t* sem_P_fromSupplier, sem_t* sem_C_fromSupplier, sem_t* sem_D_fromSupplier, sem_t* sem_P_toCounter, sem_t* sem_C_toCounter, sem_t* sem_D_toCounter, sem_t* sizeofKitchen, sem_t* mutexKitchen, sem_t* mutexCook, sem_t* mutexCounter, sem_t* counterSize, sem_t* counterSoupSize, sem_t* counterMainCourseSize, sem_t* counterDesertSize, int size, int N);

sem_P_fromSupplier --> The number of soups in the kitchen
sem_C_fromSupplier --> The number of main courses in the kitchen
sem_D_fromSupplier --> The number of deserts in the kitchen
sem_P_toCounter --> Total number of soups taken to the counter
sem_C_toCounter --> Total number of main courses taken to the counter

sem_D_toCounter --> Total number of deserts taken to the counter
sizeofKitchen --> The number of free spaces in the kitchen is initially 2 * L * M + 1.
mutexKitchen --> It will be used to prevent both supplier and cook from entering the kitchen at the same time.
mutexCook --> Multiple cooks should be prevented from entering the kitchen at the same time.
mutexCounter --> Leaving food on the counter or taking it at the same time should be prevented.
counterSize --> The number of free plates in the counter
counterSoupSize --> At that moment, the number of soups on the counter
counterMainCourseSize --> At that moment, the number of main courses on the counter
counterDesertSize --> At that moment, the number of deserts on the counter
size --> L * M
N --> Total number of cooks

int theGraduated(sem_t* counterSoupSize, sem_t* counterMainCourseSize, sem_t* counterDesertSize, sem_t* counterSize, sem_t* mutexCounter, sem_t* mutexStudentsGraduated, sem_t* emptyTable, sem_t* mutexGraduated, sem_t* totalStudents, int limit, int G);

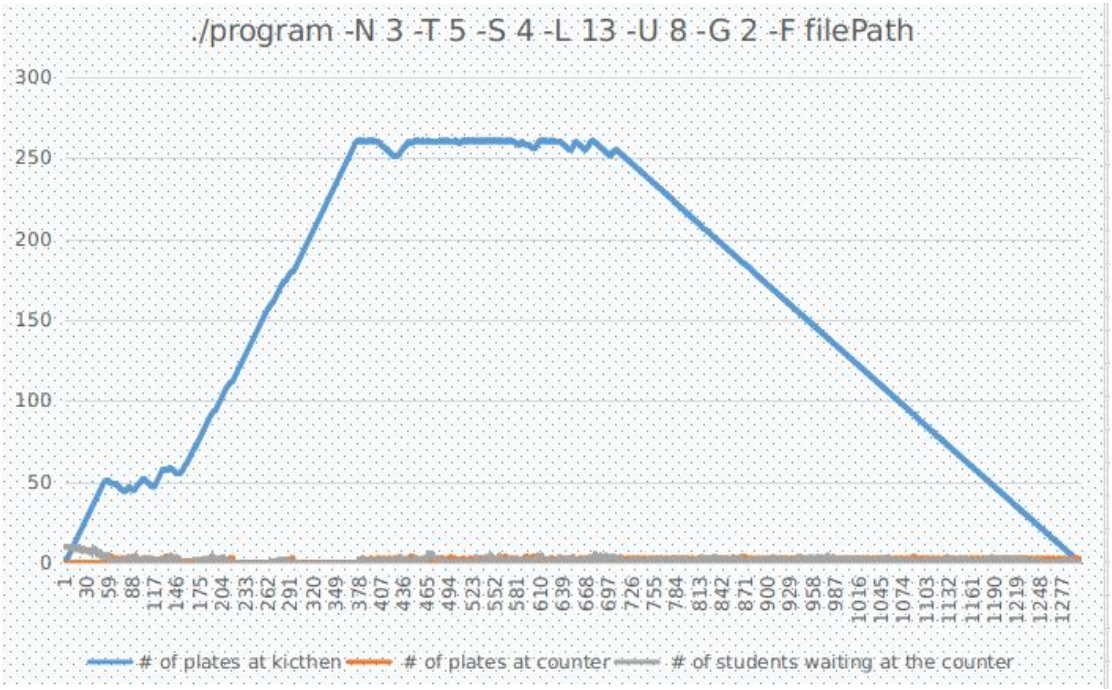
int theStudent(sem_t* counterSoupSize, sem_t* counterMainCourseSize, sem_t* counterDesertSize, sem_t* counterSize, sem_t* mutexCounter, sem_t* mutexStudents, sem_t* emptyTable, sem_t* mutexGraduated, sem_t* totalStudents, int limit, int U);

counterSoupSize --> At that moment, the number of soups on the counter
counterMainCourseSize --> At that moment, the number of main courses on the counter
counterDesertSize --> At that moment, the number of deserts on the counter
counterSize --> The number of free plates in the counter
mutexCounter --> Leaving food on the counter or taking it at the same time should be prevented.
emptyTable --> The number of empty tables is kept
mutexGraduated --> Graduated students will eat first
limit --> L parameter
G or U --> Number of students graduating or normal students

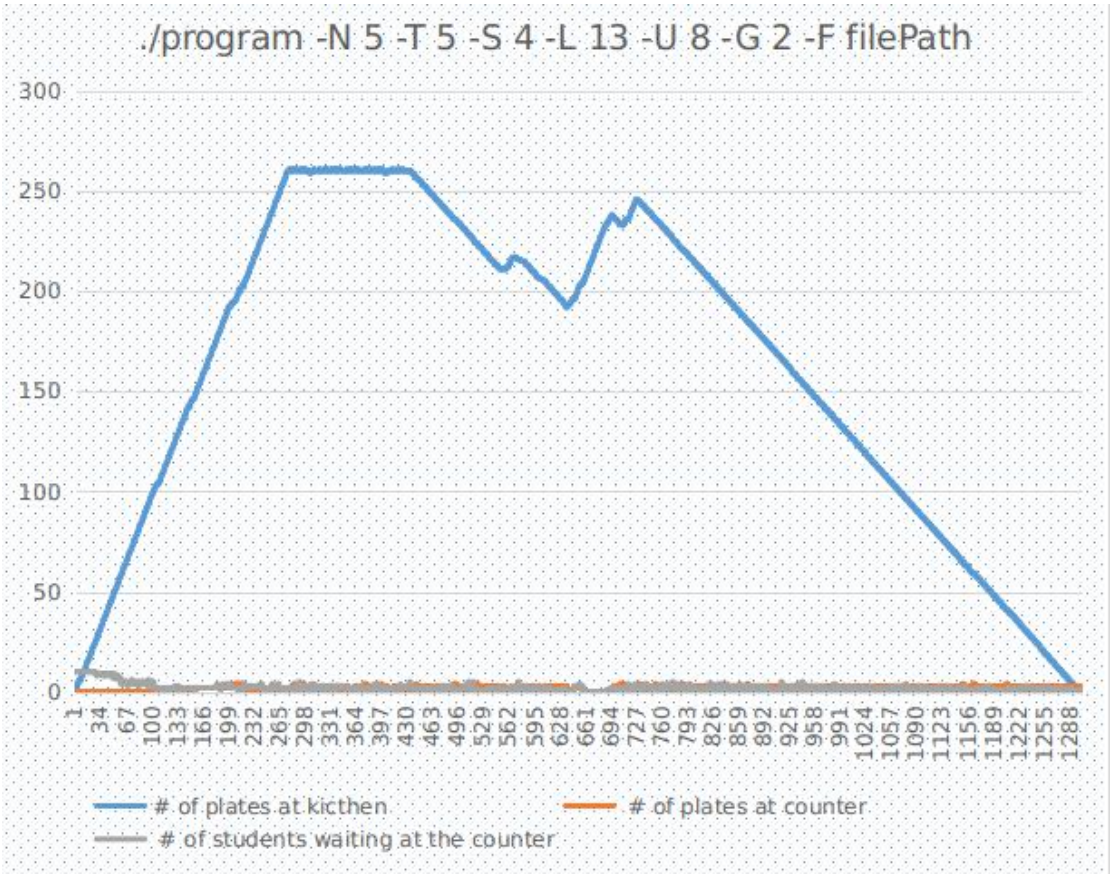
The graphic plots were tested by writing to the file as PCDPCDPCDPCD format.

GRAPHICAL PLOTS :

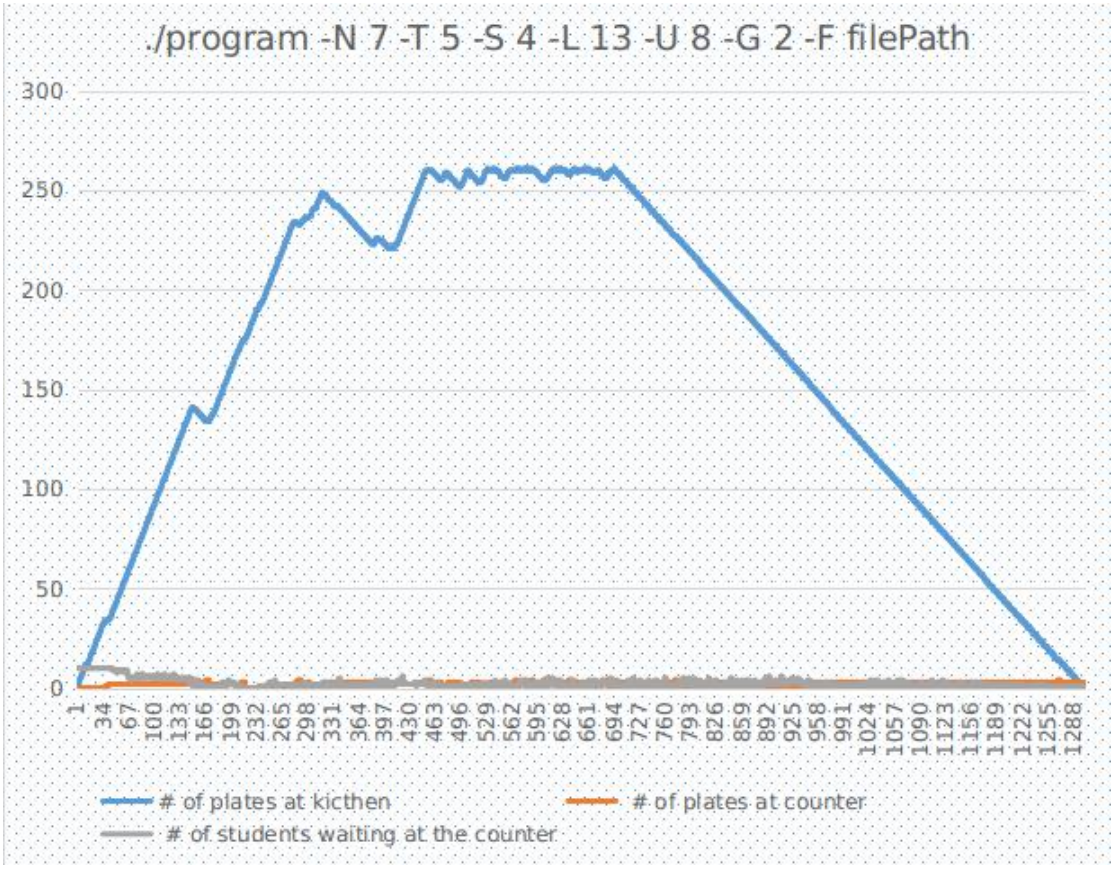
A) Changed N Parameter



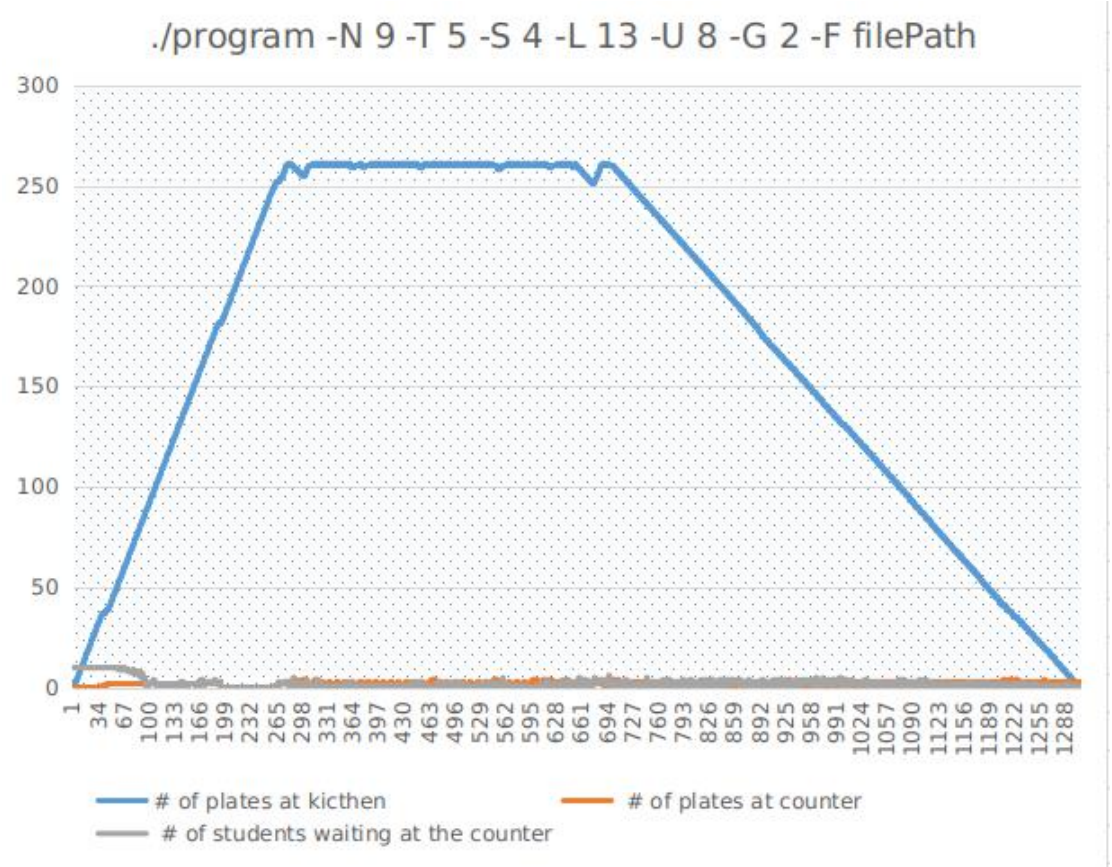
N = 3



N = 5

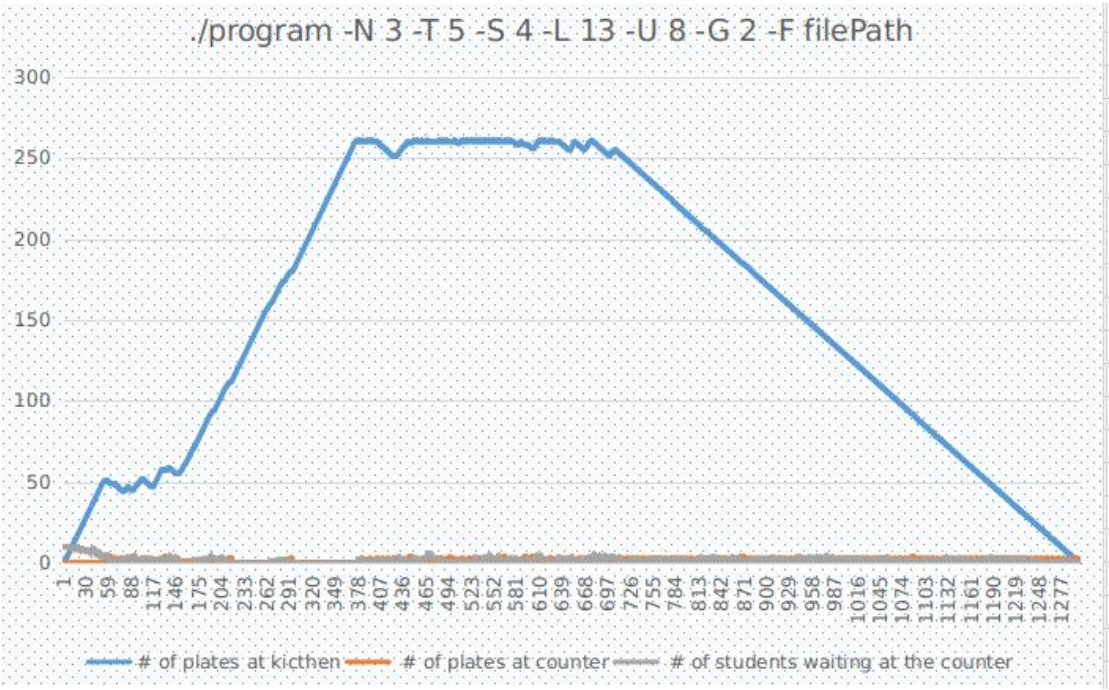

./program -N 7 -T 5 -S 4 -L 13 -U 8 -G 2 -F filePath
300
250
200
150
100
50
0
1
34
67
100
133
166
199
232
265
298
331
364
397
430
463
496
529
562
595
628
661
694
727
760
793
826
859
892
925
958
991
1024
1057
1090
1123
1156
1189
1222
1255
1288
# of plates at kicthen
# of plates at counter
# of students waiting at the counter

N = 7

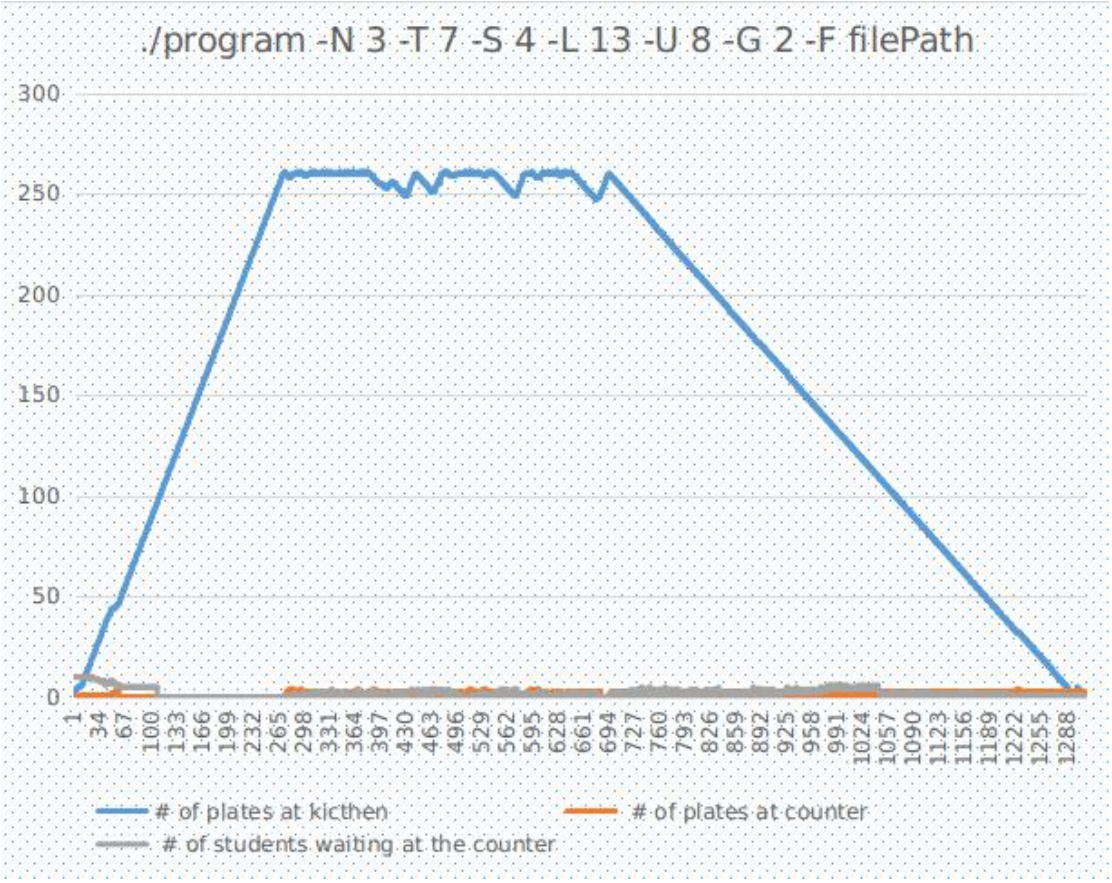

./program -N 9 -T 5 -S 4 -L 13 -U 8 -G 2 -F filePath
300
250
200
150
100
50
0
1
34
67
100
133
166
199
232
265
298
331
364
397
430
463
496
529
562
595
628
661
694
727
760
793
826
859
892
925
958
991
1024
1057
1090
1123
1156
1189
1222
1255
1288
# of plates at kicthen
# of plates at counter
# of students waiting at the counter

N = 9

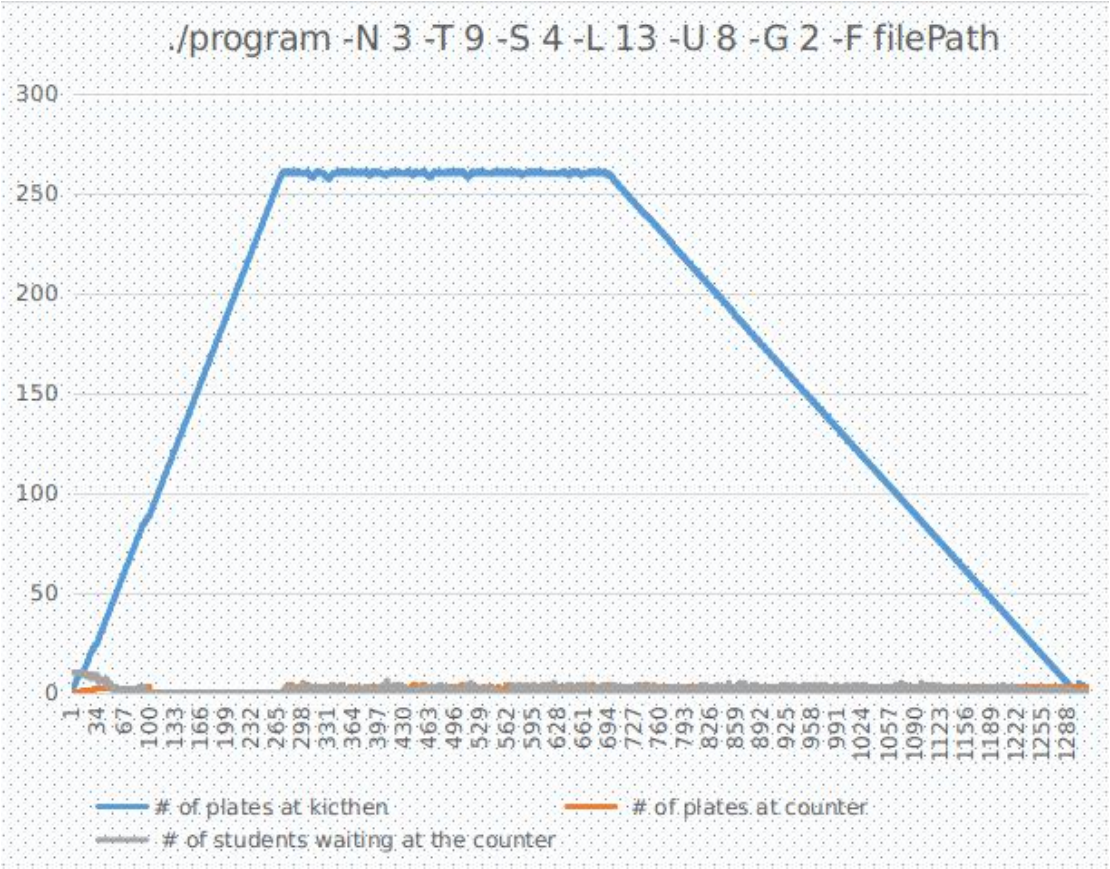
B) Changed T Parameter



T = 5

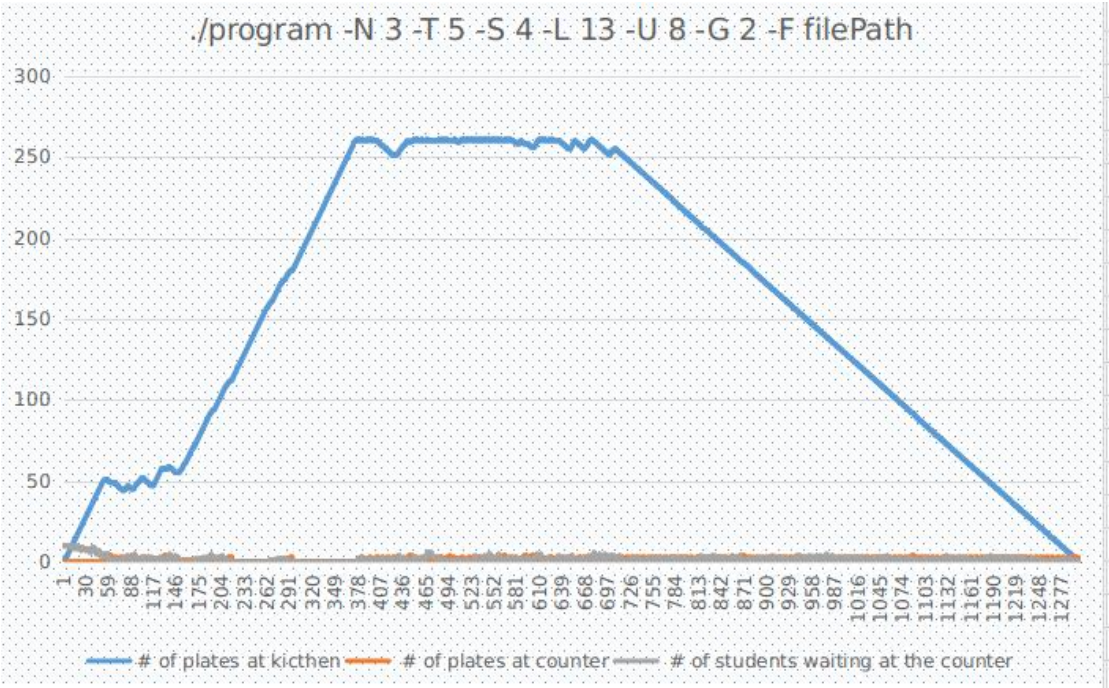


T = 7

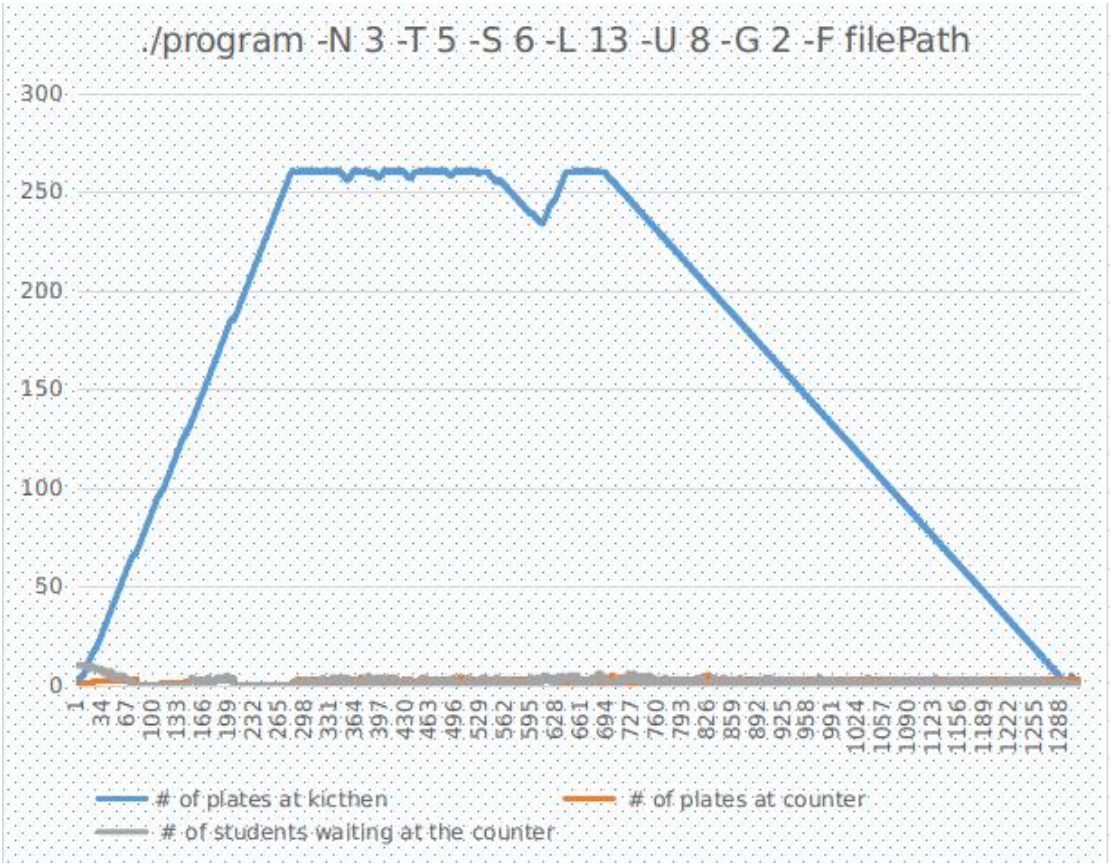


T = 9

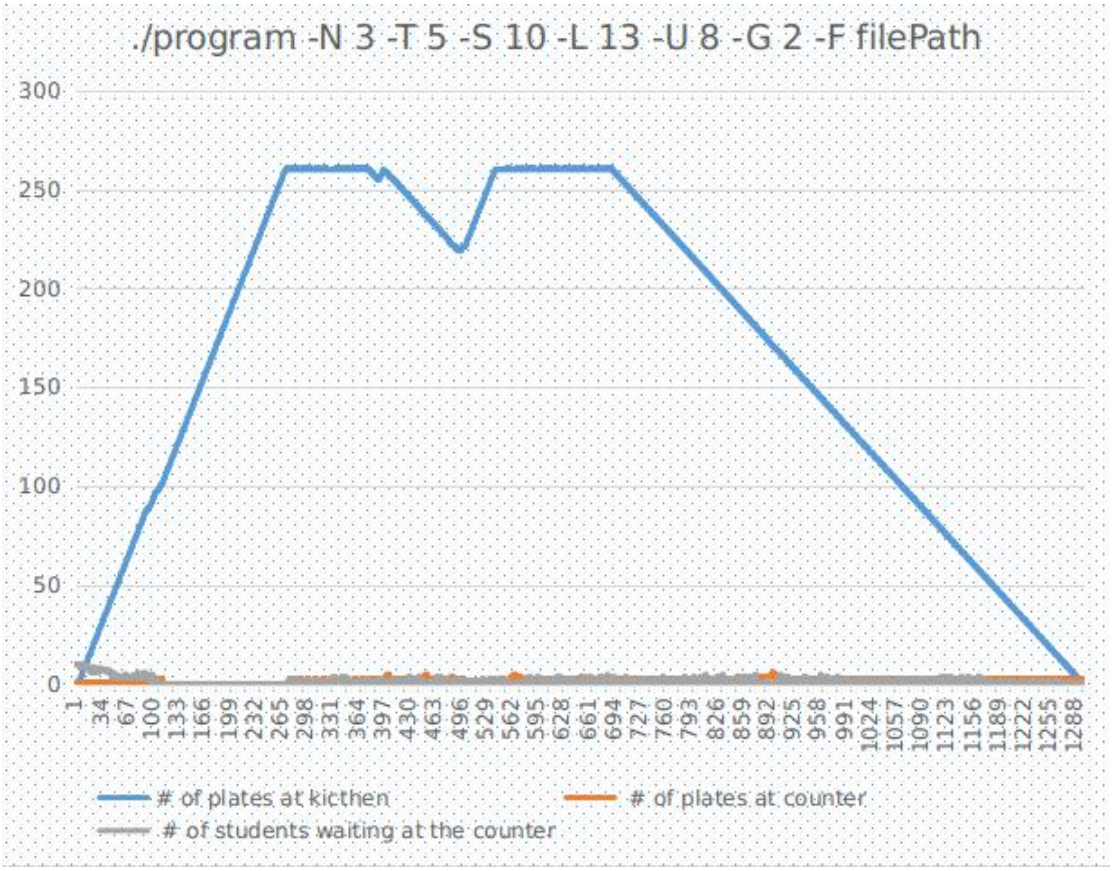
C) Changed S Parameter



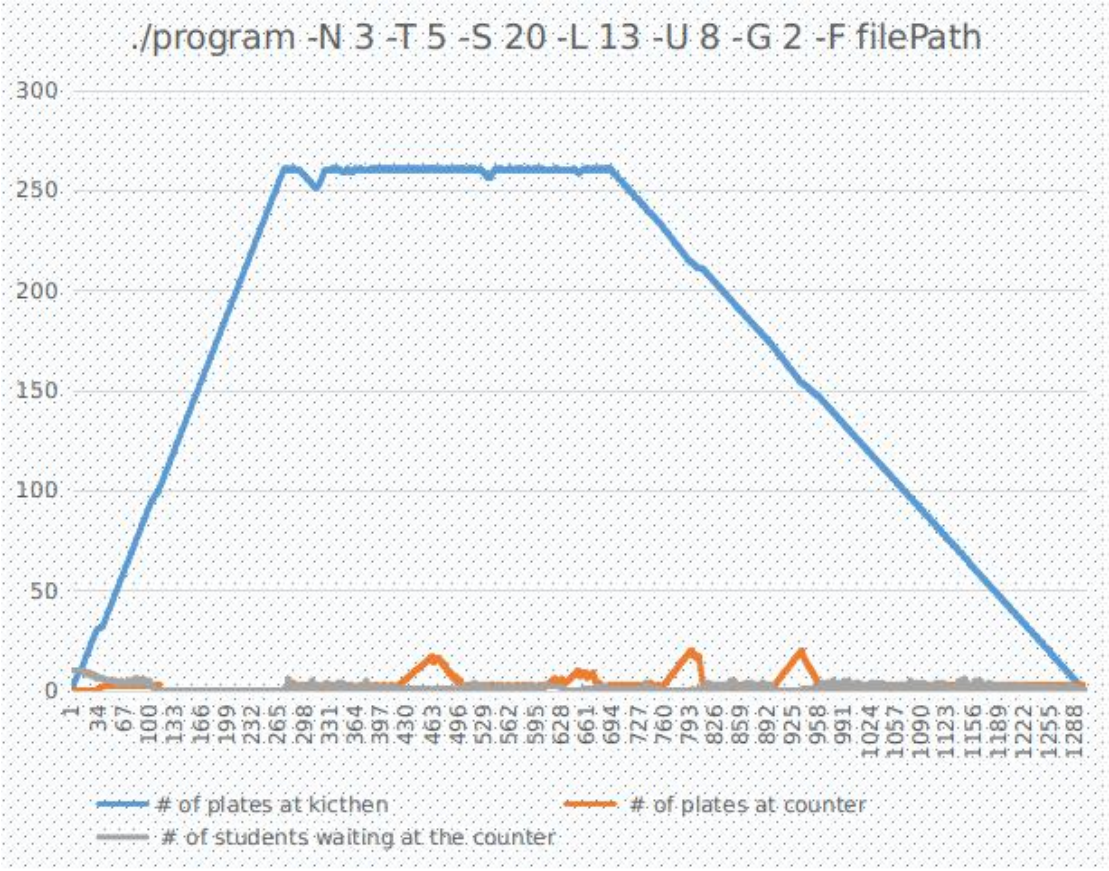
S = 4


./program -N 3 -T 5 -S 6 -L 13 -U 8 -G 2 -F filePath
300
250
200
150
100
50
0
1
34
67
100
133
166
199
232
265
298
331
364
397
430
463
496
529
562
595
628
661
694
727
760
793
826
859
892
925
958
991
1024
1057
1090
1123
1156
1189
1222
1255
1288
# of plates at kicthen
# of plates at counter
# of students waiting at the counter

S = 6

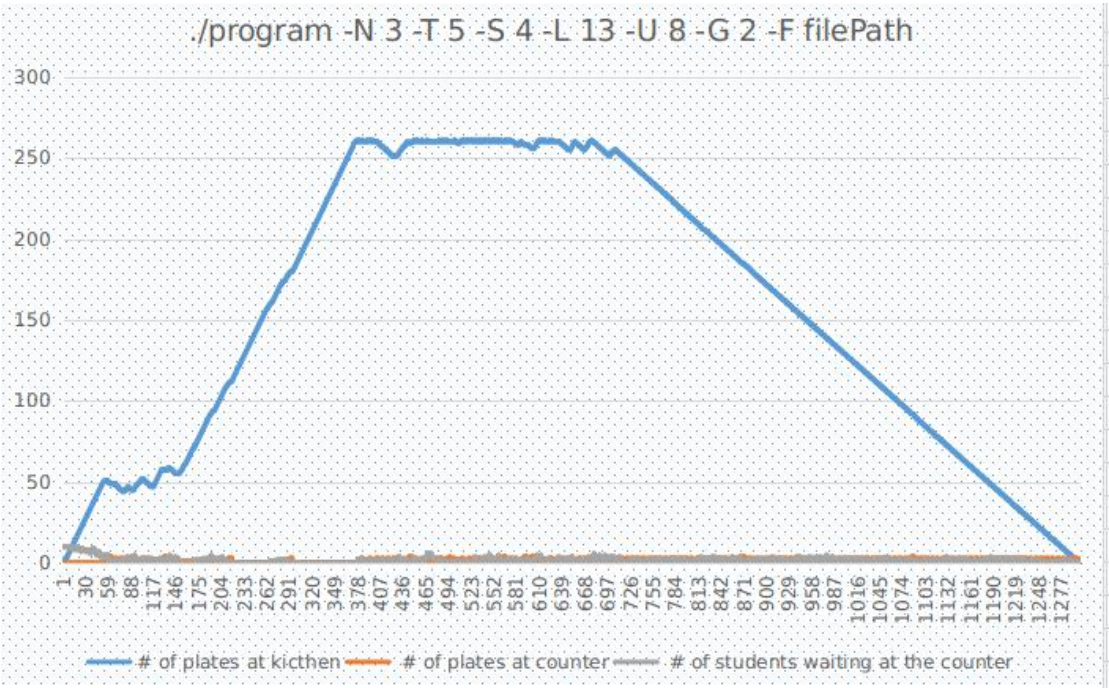

./program -N 3 -T 5 -S 10 -L 13 -U 8 -G 2 -F filePath
300
250
200
150
100
50
0
1
34
67
100
133
166
199
232
265
298
331
364
397
430
463
496
529
562
595
628
661
694
727
760
793
826
859
892
925
958
991
1024
1057
1090
1123
1156
1189
1222
1255
1288
# of plates at kicthen
# of plates at counter
# of students waiting at the counter

S = 10

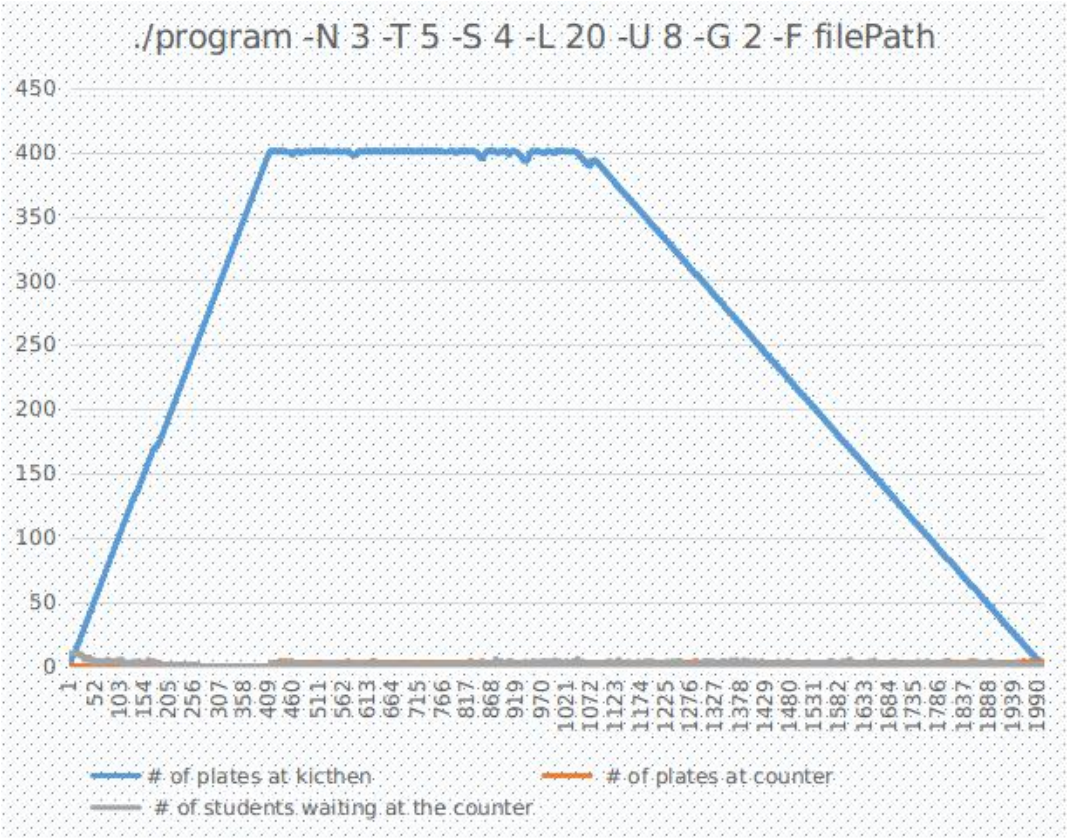


S = 20

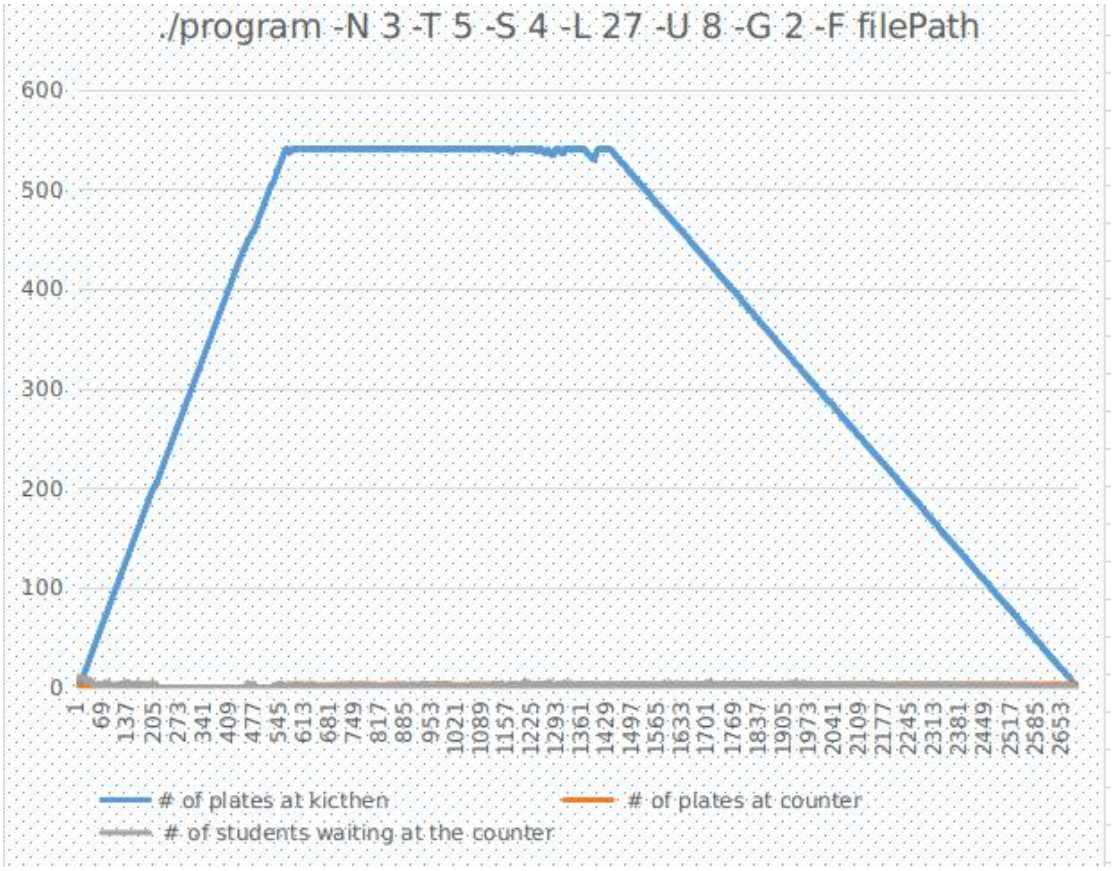
D) Changed L Parameter



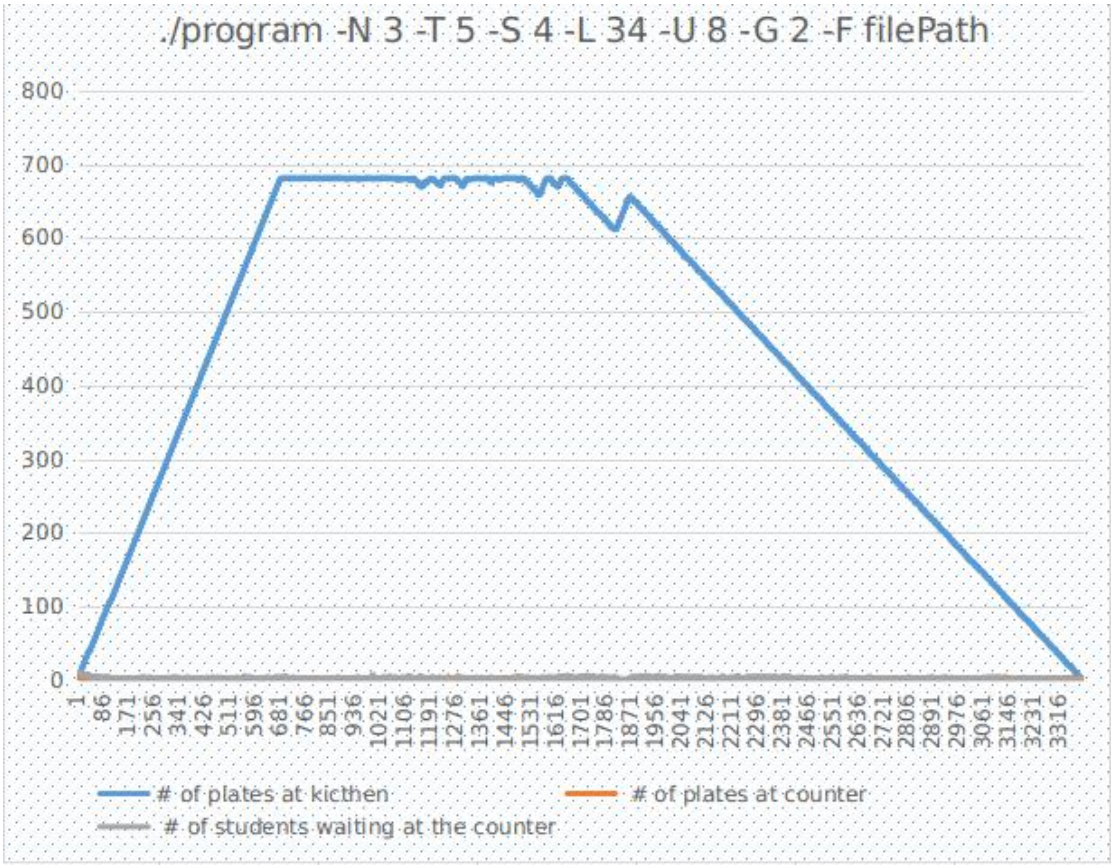
L = 13


./program -N 3 -T 5 -S 4 -L 20 -U 8 -G 2 -F filePath
450
400
350
300
250
200
150
100
50
0
1
52
103
154
205
256
307
358
409
460
511
562
613
664
715
766
817
868
919
970
1021
1072
1123
1174
1225
1276
1327
1378
1429
1480
1531
1582
1633
1684
1735
1786
1837
1888
1939
1990
# of plates at kicthen
# of plates at counter
# of students waiting at the counter

L = 20

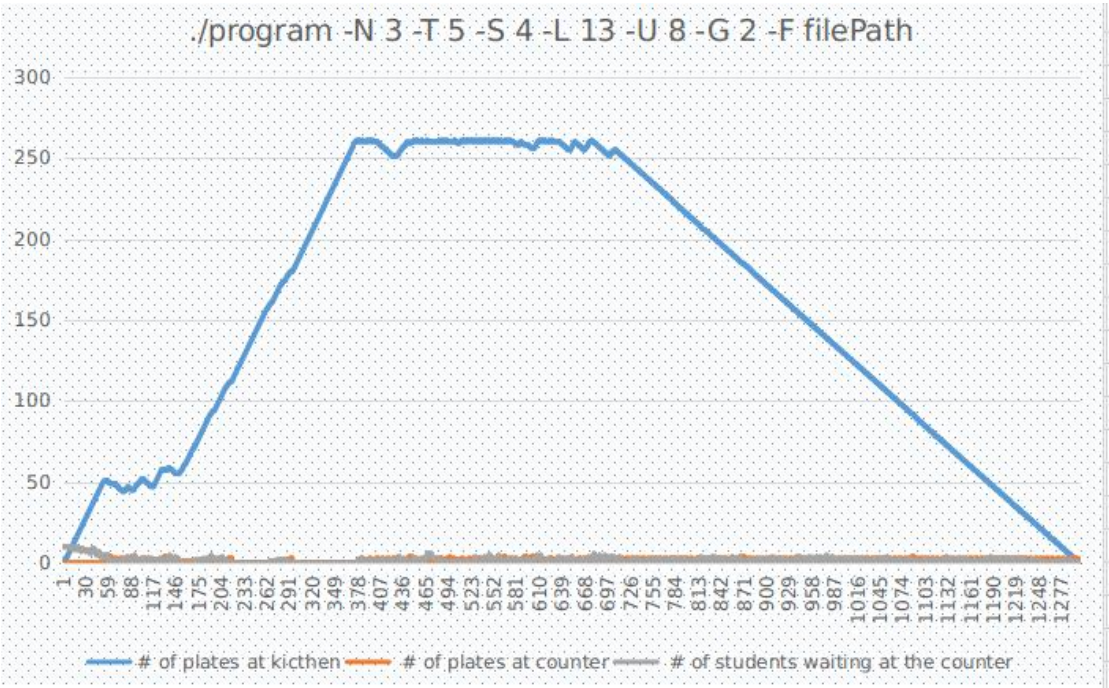

./program -N 3 -T 5 -S 4 -L 27 -U 8 -G 2 -F filePath
600
500
400
300
200
100
0
1
69
137
205
273
341
409
477
545
613
681
749
817
885
953
1021
1089
1157
1225
1293
1361
1429
1497
1565
1633
1701
1769
1837
1905
1973
2041
2109
2177
2245
2313
2381
2449
2517
2585
2653
# of plates at kicthen
# of plates at counter
# of students waiting at the counter

L = 27

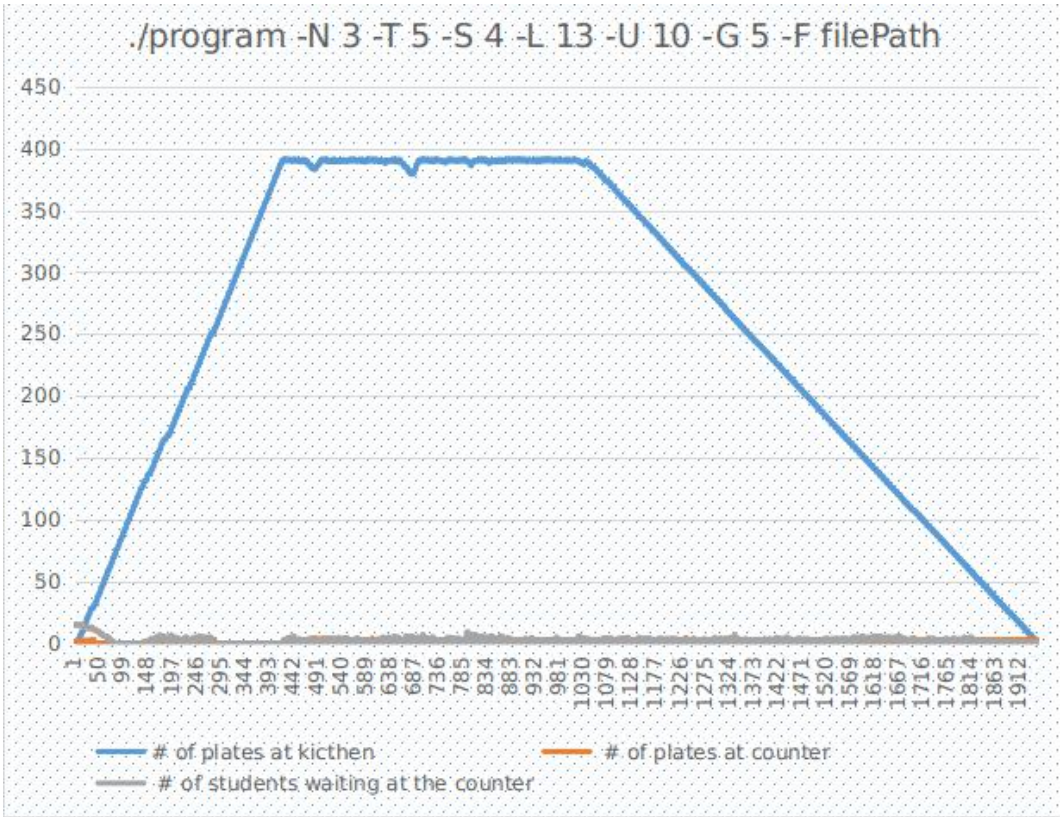


L = 34

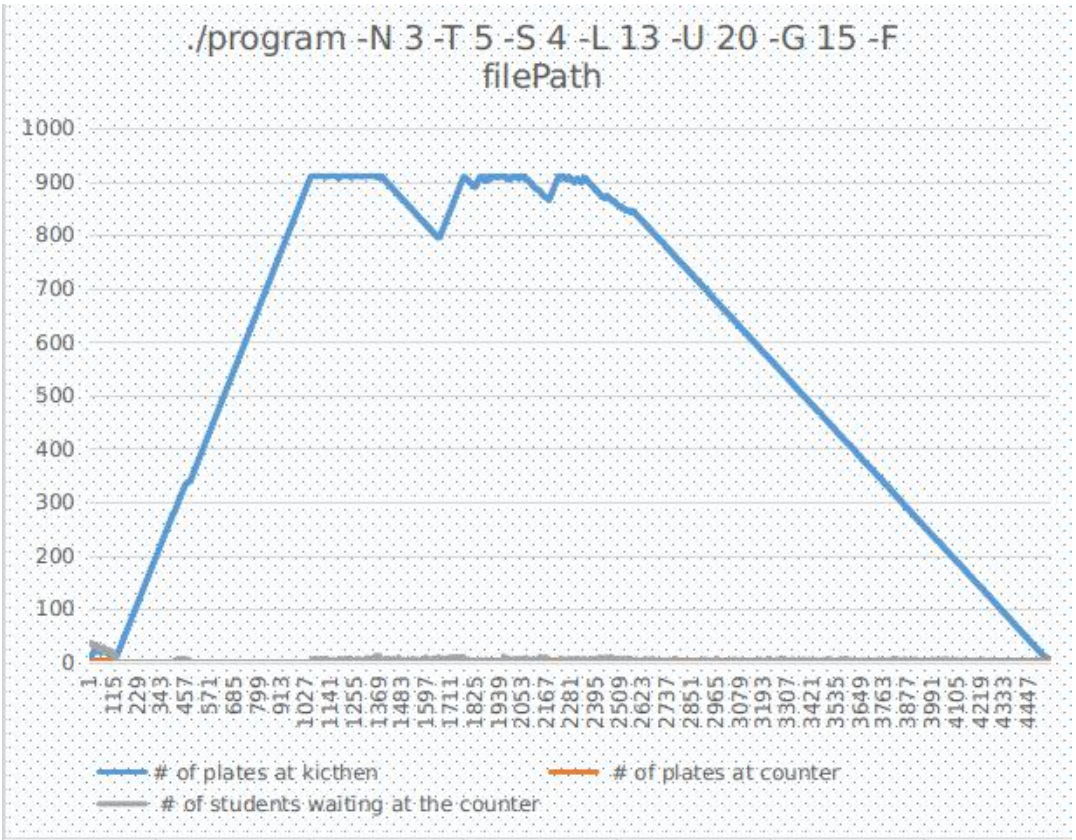
E) Changed M Parameter



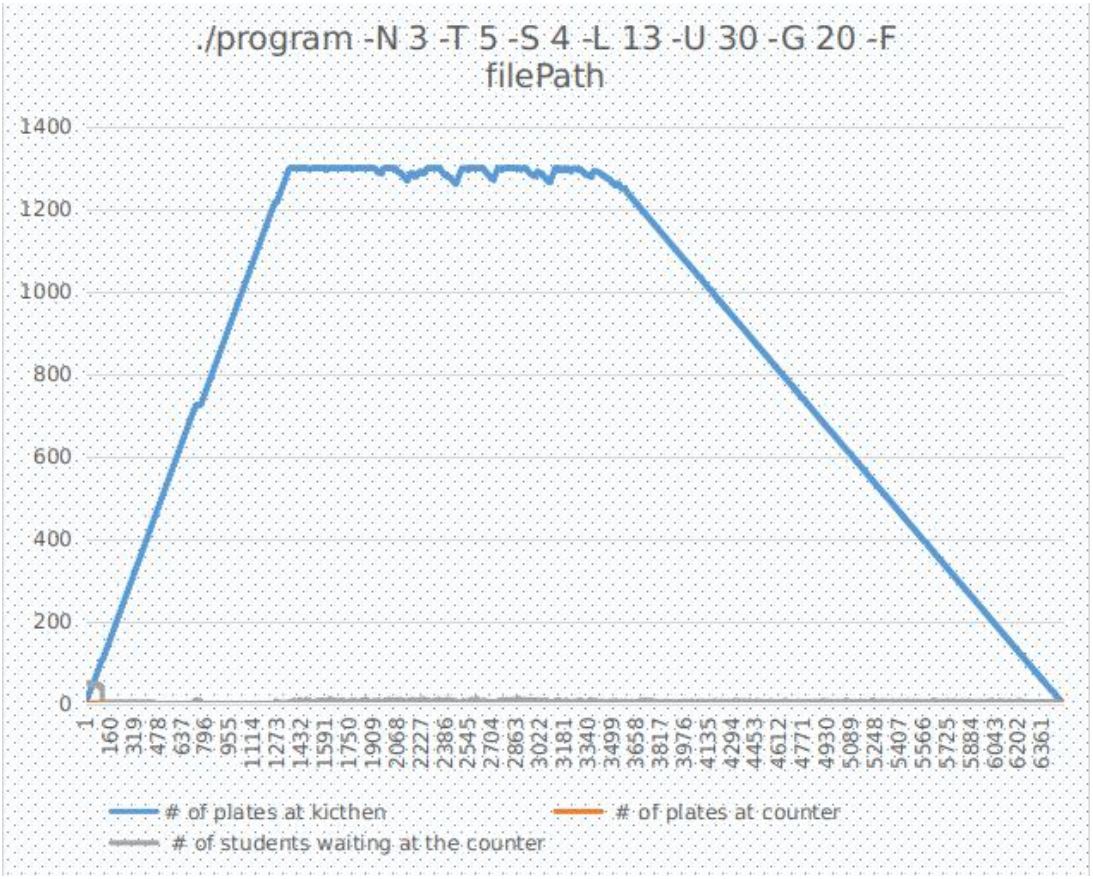
M = 10 (U = 8 and G = 2)



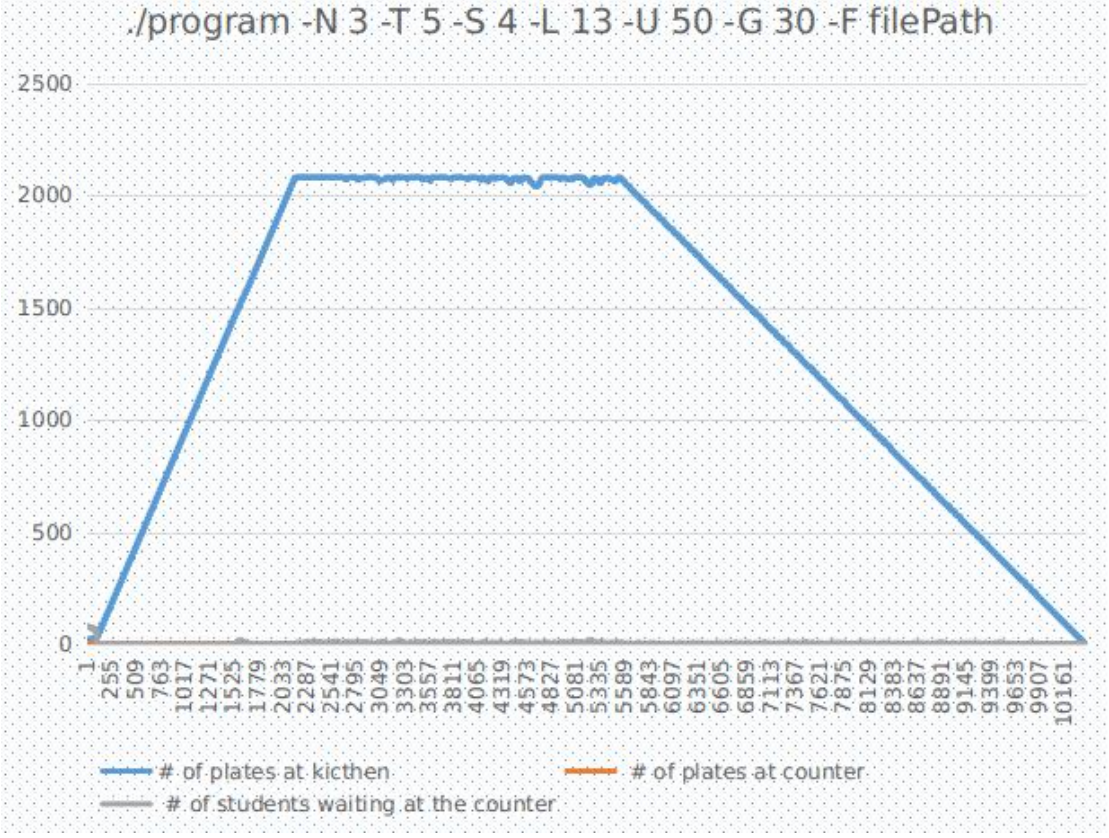
M = 15 (U = 10 and G = 5)



M = 35 (U = 20 and G = 5)



M = 50 (U = 30 and G = 20)



M = 80 (U = 50 and G = 30)