CS342 – Operating Systems

Project 3

Section 2

Barış Ardıç

21401578

Description of The Solution :

I will describe what I did chronologically while looking my source code therefore you can follow what's going on there as well.

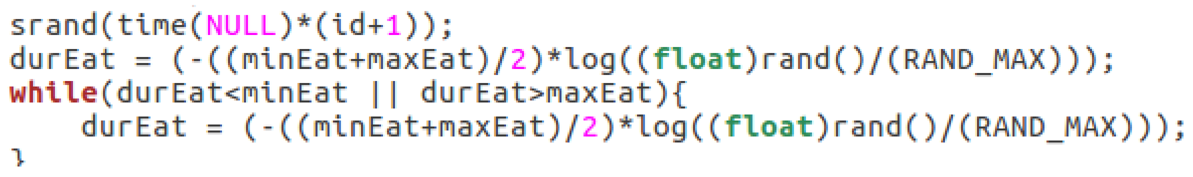
- I declare my variables for the parameters and mutexes
- In the main I take the parameters from argv[] to my initialized variables and declared arrays for chopsticks and philosophers.
- I initialize create and join thread in loops.
- Rest of the code is the runner function of threads. (philosophers)
- I simulated left and right forks of philosophers using %(#PHILOS) then I go into the while(#count--) starting by making thread id=1 sleep for 1ms thus I prevent the deadlock case where all philosophers take their right fork.
- Threads are in the hungry state when they start(they did not acquired forks to eat yet) so I measure time until both forks are acquired. That's hungry time.
- Then I check for distribution and I create random variates accordingly

  For uniform distribution, I use the following method:

  ```
  durThink = maxThink + rand()/(RAND_MAX/(minThink-maxThink+1)+1);
  ```

  Rand()/Rand_MAX here gives a uniform random value between (0,1) interval. Rest adjusts the value for our min mac parameters, eat duration also uses this if parameter is uniform

For exponential distribution I use :



```
srand(time(NULL)*(id+1));
durEat = (-((minEat+maxEat)/2)*log((float)rand()/(RAND_MAX)));
while(durEat<minEat || durEat>maxEat){
    durEat = (-((minEat+maxEat)/2)*log((float)rand()/(RAND_MAX)));
}
```

durEat is essentially this inverse function of cumulative exponential distribution.(This is the most widely used method in generating exponential variates and it is explained in exponential distribution Wikipedia page.)

$$T = \frac{-\ln(U)}{\lambda}.$$

For us T here is durEat lambda is 1/mean and U is again uniform random value between (0,1). If our variate is not in our range from program parameters we generate it again. I thought this would slow down the program however I works really well there is almost none measurable runtime difference.

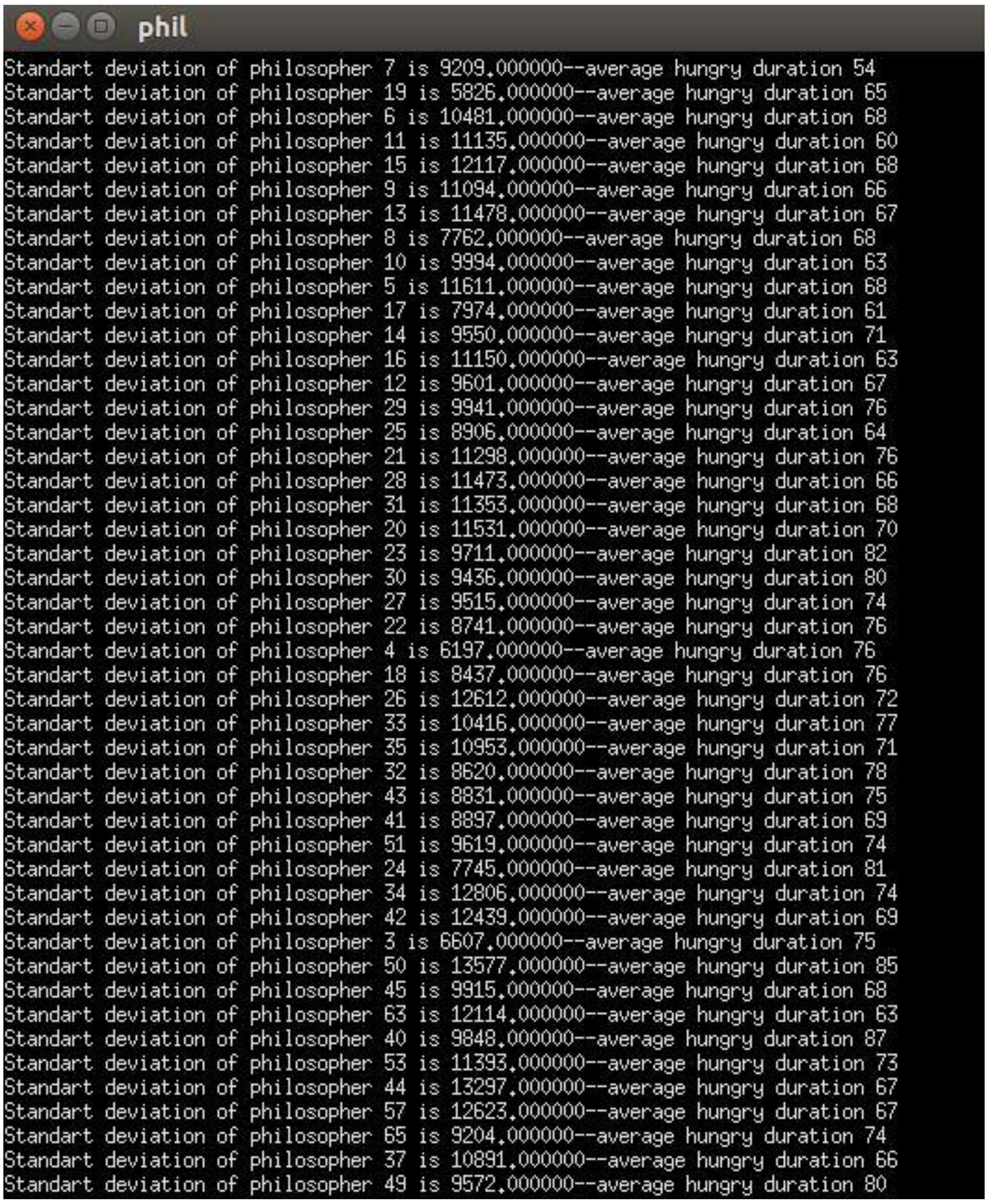
- After our think and eat durations are generated we eat think then we iterate the while(count--) until we meet count.
- I managed to get different values from rand() by seeding time(NULL*(id+1)) to srand(). With only time(NULL) I used to get the same random number for different threads while they are eating/thinking. Because the system clock time() uses was cast to integer for variate when rand() is called microsecond after I would get the same value. I avoid this by multiplying by (id+1)

Notes :

- There are commented section in my code they were for measuring standard deviation and average for think process.
- I talked with Özcan Hoca and he was also positive on this idea : There should not be a test case where there is only 1 philosopher since it is against the problem description. With 1 philosophers, there is only 1 fork. The whole problem is based on philosopher not being able to eat with 1 fork because otherwise we would have no deadlock case.(The case which all philosophers the their right fork.)

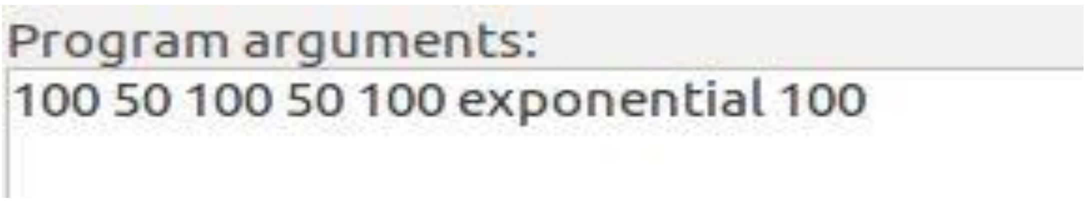
Standard Deviation and Average of Think

- As I mentioned the code which did this calculation are still in my source they are just commented out. Here are some pictures of example runs that I did.



```
Standart deviation of philosopher 7 is 9209.000000--average hungry duration 54
Standart deviation of philosopher 19 is 5826.000000--average hungry duration 65
Standart deviation of philosopher 6 is 10481.000000--average hungry duration 68
Standart deviation of philosopher 11 is 11135.000000--average hungry duration 60
Standart deviation of philosopher 15 is 12117.000000--average hungry duration 68
Standart deviation of philosopher 9 is 11094.000000--average hungry duration 66
Standart deviation of philosopher 13 is 11478.000000--average hungry duration 67
Standart deviation of philosopher 8 is 7762.000000--average hungry duration 68
Standart deviation of philosopher 10 is 9994.000000--average hungry duration 63
Standart deviation of philosopher 5 is 11611.000000--average hungry duration 68
Standart deviation of philosopher 17 is 7974.000000--average hungry duration 61
Standart deviation of philosopher 14 is 9550.000000--average hungry duration 71
Standart deviation of philosopher 16 is 11150.000000--average hungry duration 63
Standart deviation of philosopher 12 is 9601.000000--average hungry duration 67
Standart deviation of philosopher 29 is 9941.000000--average hungry duration 76
Standart deviation of philosopher 25 is 8906.000000--average hungry duration 64
Standart deviation of philosopher 21 is 11298.000000--average hungry duration 76
Standart deviation of philosopher 28 is 11473.000000--average hungry duration 66
Standart deviation of philosopher 31 is 11353.000000--average hungry duration 68
Standart deviation of philosopher 20 is 11531.000000--average hungry duration 70
Standart deviation of philosopher 23 is 9711.000000--average hungry duration 82
Standart deviation of philosopher 30 is 9436.000000--average hungry duration 80
Standart deviation of philosopher 27 is 9515.000000--average hungry duration 74
Standart deviation of philosopher 22 is 8741.000000--average hungry duration 76
Standart deviation of philosopher 4 is 6197.000000--average hungry duration 76
Standart deviation of philosopher 18 is 8437.000000--average hungry duration 76
Standart deviation of philosopher 26 is 12612.000000--average hungry duration 72
Standart deviation of philosopher 33 is 10416.000000--average hungry duration 77
Standart deviation of philosopher 35 is 10953.000000--average hungry duration 71
Standart deviation of philosopher 32 is 8620.000000--average hungry duration 78
Standart deviation of philosopher 43 is 8831.000000--average hungry duration 75
Standart deviation of philosopher 41 is 8897.000000--average hungry duration 69
Standart deviation of philosopher 51 is 9619.000000--average hungry duration 74
Standart deviation of philosopher 24 is 7745.000000--average hungry duration 81
Standart deviation of philosopher 34 is 12806.000000--average hungry duration 74
Standart deviation of philosopher 42 is 12439.000000--average hungry duration 69
Standart deviation of philosopher 3 is 6607.000000--average hungry duration 75
Standart deviation of philosopher 50 is 13577.000000--average hungry duration 85
Standart deviation of philosopher 45 is 9915.000000--average hungry duration 68
Standart deviation of philosopher 63 is 12114.000000--average hungry duration 63
Standart deviation of philosopher 40 is 9848.000000--average hungry duration 87
Standart deviation of philosopher 53 is 11393.000000--average hungry duration 73
Standart deviation of philosopher 44 is 13297.000000--average hungry duration 67
Standart deviation of philosopher 57 is 12623.000000--average hungry duration 67
Standart deviation of philosopher 65 is 9204.000000--average hungry duration 74
Standart deviation of philosopher 37 is 10891.000000--average hungry duration 66
Standart deviation of philosopher 49 is 9572.000000--average hungry duration 80
```

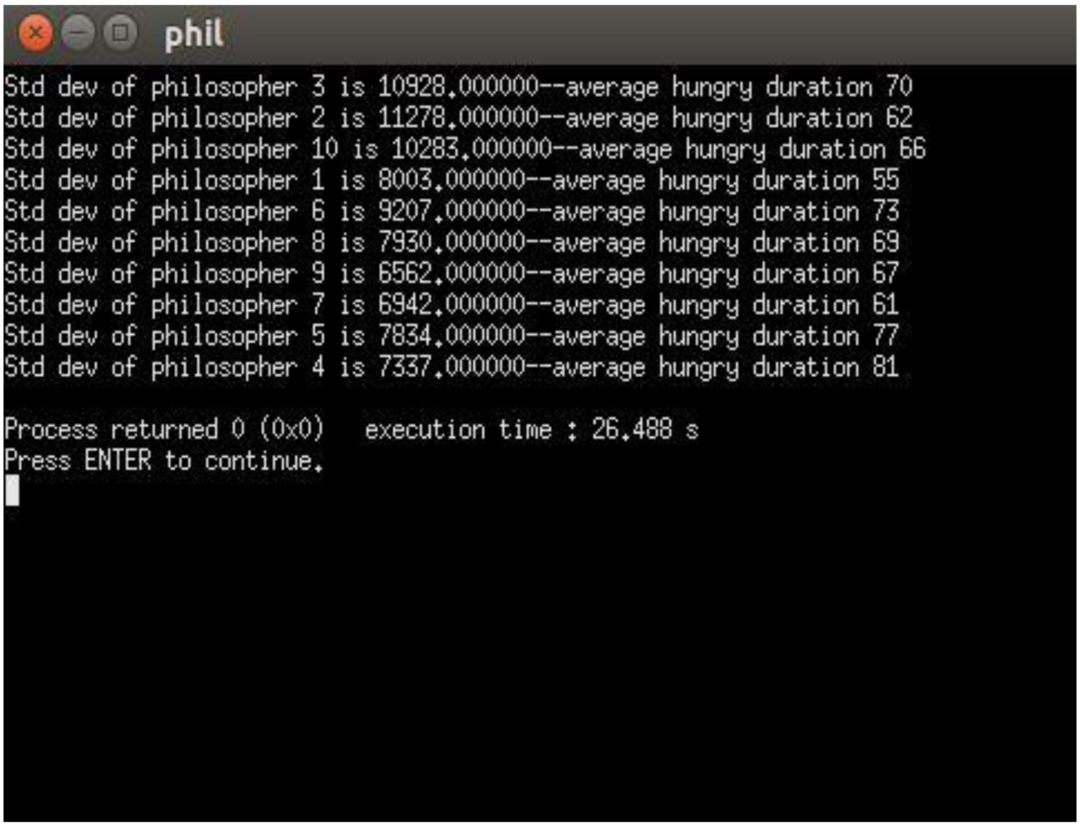
These are the program parameters(I was using codeblocks, not terminal thus UI might seem odd**)**



Program arguments:

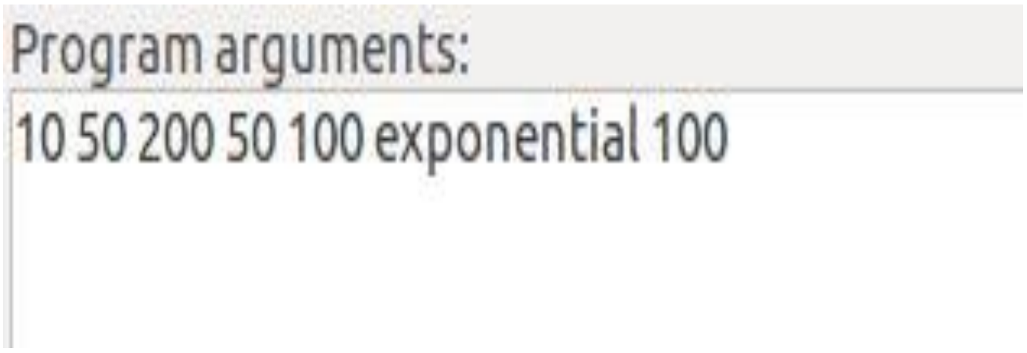
```
100 50 100 50 100 exponential 100
```

Another Run:



```
phil
Std dev of philosopher 3 is 10928.000000--average hungry duration 70
Std dev of philosopher 2 is 11278.000000--average hungry duration 62
Std dev of philosopher 10 is 10283.000000--average hungry duration 66
Std dev of philosopher 1 is 8003.000000--average hungry duration 55
Std dev of philosopher 6 is 9207.000000--average hungry duration 73
Std dev of philosopher 8 is 7930.000000--average hungry duration 69
Std dev of philosopher 9 is 6562.000000--average hungry duration 67
Std dev of philosopher 7 is 6942.000000--average hungry duration 61
Std dev of philosopher 5 is 7834.000000--average hungry duration 77
Std dev of philosopher 4 is 7337.000000--average hungry duration 81

Process returned 0 (0x0)   execution time : 26.488 s
Press ENTER to continue.
```



Program arguments:

```
10 50 200 50 100 exponential 100
```