Programing Assignment 1 Report

The files included are a README, this report, and 2 .cpp files. The README shows the commands required to compile and run the 2 .cpp files. The 2 .cpp files pertain to each of the questions asked in the assignment. “p1.cpp” refers to problem 1 where it generates a workload based on a Poisson distribution with a wavelength of 2 and a service time based on an exponential distribution with a wavelength of 1. “p1.cpp” also calculates the actual average rate and actual average service time based on the results given by the simulated workload. “p2.cpp” refers to problem 2 which involves generating reports based on two identical servers with a MTBF of 500 hours, a 10 hour repair period and an exponential distribution across 20 years. “p2.cpp” also generates a total failure based on when both of the servers are down at the same time for the first time.

Problem 1:

For “p1.cpp” it prints 1000 processes with their process ids, arrival times in seconds, and service time requested in seconds. The processes are structs stored in an array then displayed onto the screen. They follow the format provided which is:

<process id, arrival time, service time requested>

A picture containing timeline

Description automatically generated

The process ids were generated incrementally as more processes were added to the array.

The arrival times are generated using the following:



This gets the arrival times based on the sum of arr which starts at zero, and grows based on the equation “–(1/2)\*ln(r1)”. “r1” generates a uniform distribution through “rand()/RAND\_MAX;” and log(r1) = ln(r1). This equation generates the arrivals based on a Poisson distribution.

The service times were generated using the following: 

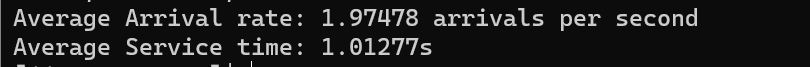
“-1.0\*log(r1)” represents the generation of service times following an exponential distribution.

The program also generated the actual average process per second and average service time based on the results from the current generation.



Since the program’s seed uses time(0), it uses different seeds each time it is run.





Which produces averages 1.96 arrivals per second and 1.02s, which is close to the averages provided.

Problem 2:

“p2.cpp” compares 2 servers that mirror each other and have the exact same MTBF of 500. “p2.cpp” creates an array of structs that contain all of the failure and restoration times a server has based on the hours they went down and were restored.



These arrays were generated through the “server()” function which takes a server struct array and populates it using:



Which creates an exponential distribution of failure times across the 20 years. The restored times are then calculated by adding 10.0 to the failure times. The “server()” function also returns the size of the array. “p2.cpp” contains the “mirror()” function which calls the server function twice to generate 2 identical servers and compares them using the “total\_failure()” function, which takes both server arrays and the smallest array size and returns when both servers are down at the same time for the first time.



This value is also returned by the “mirror()” function.

“p2.cpp” also generates this scenario 5 times with 5 different seeds and averages them out based on the 5 seeds used.

