UMass Boston CS 444
Homework 4 - Background for proj4
Posted Tuesday, December 2, 2025
Due by Thursday, December 4, 2025 at 8pm if possible
But by Saturday, December 6, 2025 at 11:59pm for sure
And after the majority of you have submitted it, q.c, will be posted

1 Overview

Homework may be handwritten on this form. Late submission gets the late penalty. To submit your homework, prepare a file called `hw4.txt` — the filename must be exactly `hw4.txt`. Upload the file to the `cs444` folder in your home directory on the CS Linux server. Do not place the file in a subdirectory or it will not be collected by the script.

This hw4 looks at the parts of proj4 that you need to understand well. Best to do this ahead of being given the code that you will copy and modify.

1.1 Login to CS Server

Begin by logging in to the CS server from your laptop. Type: `w|more followed by ENTER` to see others who are logged in and what they are working on. You need to actually be working on the server to see your name on the list. If you are on VSCode, please exit it and login actually. Write the complete line of data given by command "w" for your login.
ANS:

1.2 Make subdirectory `proj4`

You are first asked to make a subdirectory `proj4` in your course directory. Type: `mkdir proj4`. Then display it: `ls -la`. Write the full line of information for the proj4 directory including permissions.
ANS:

1.3 Copy in supplied files

Next you are asked to copy in all the supplied files. Use `cp -r /home/hdeblois/cs444/proj4/f25/ .`. The "-r" is for a recursive copy. The destination is "." which means same name as what was given. Type: `ls -la` to see all the files with bytecounts. How many files do you have?
ANS:

2 Getting Started

2.1 Look at a threaded program: hello32.c

Next display the C file: `hello32.c` from the tutorial given in the posixthreads slides. Type: `cat hello32.c|more`. Use ENTER to continue down the page. How are the threads terminated? How are the threads created?

ANS: terminated by:

ANS: created by:

2.2 Compile and Run

Then look at the supplied file with the compile command `cat How_to_compile_C_code_with_pthread_dot_h`. Then compile hello32.c and run. Were the threads run in order?

ANS:

2.3 Change the number of threads to 10

Make the change and recompile. Then run. Record the order in which the threads completed. List all 10.

ANS:

2.4 Run Instructor's simulation

Next you are asked to run the given instructor's basic `simulationhd`. Type: `./simulationjhd`. Your are running with default values for customer arrival rate lambda and service rate mu. The number of customers and number of servers are also given. List the parameter output that is printed at the top of the screen.

ANS:

2.5 Watch the queue

: Watch the queue display. (Note that the code runs the command `clear` to clear the screen so the output appears at the top. This will also clear any previous display, so remember to place your debug outputs carefully to avoid being erased.) Then the queue of customers is displayed. You will see a "1" for the first arrival. When the customer has been served, the number '1' will be removed. Watch until the simulation stops. What's the highest number for queue size that you saw during the run?

ANS:

2.6 Final state of display

Remember that the arrival times are randomly distributed AND distributed according to the exponential distribution. If the number in the queue increases, more numbers will be displayed. As the customers are served, numbers will be removed. After about a minute, the display will halt and the statistics will be shown. Record the final state of the queue.

ANS: Final state of queue:

2.7 Run again

: Run again. Use the same command. This time watch for the longest queue display. Record both the longest and the final state of the queue.

ANS: Longest queue seen:

ANS: Final state of queue:

2.8 Run with different input parameters

: Now plan to change an input parameter. The parameters are displayed at the top of the screen in each run. Use the switches: `-l` for lambda, `-m` for mu, `-c` for customers, `-s` for number of servers. Keep `l < m` to have a stable system. Record your full command:

ANS:

2.9 Make a different run

Play around with the parameters, and make more runs. Note what the parameter display shows, watch the queue and read the statistics. Record a changed parameter display and what you saw in the queue. Record statistics for the customer arrivals.

ANS: Parameter display:

ANS: Longest queue seen:

ANS: Final state of queue:

ANS: Interarrival time (average):

ANS: Interarrival time (standard deviation):

2.10 Statistics

: Now consider all the statistics displayed at the end of the run. But let's focus on customers. Again, the first number is the average interarrival time. The standard deviation of arrival times is also displayed. Record both:

ANS: Average interarrival time:

ANS: Standard deviation of interarrival time:

ANS: Based on your observations, describe the relationship between the queue display and the interarrival statistics as you observed it:

3 Introduction to Simulations

3.1 Queueing theory slides from class and interarrival time

There are many distributions for random variables. Look at slides 7 and 8. Lambda is the arrival rate of customers. It is expressed as arrivals per second. The formula for the interarrival time is the reciprocal. When lambda is 5 per second, what is the interarrival time? Express it as tenths of a second.

ANS:

3.2 Mean, Lambda/Mu and P_0

See queueing slide 7. What is the mean of exponential distribution, E(x)? Yes, same as the interarrival rate, 1/lambda. Next see slide 13. What are the states of the system? Yes, number of customers, starting at k=0. Note how the relationship between lambda and mu appears to control the transitions between states. Next see slide 14. Note how lambda/mu raised to the power k controls the probabilities. Let's use lambda=5, mu=10. Compute lambda/mu. Write the equation for P_0. Then compute P_0. Then compute the initial probabilities for k=1, 2, 3.

ANS: lambda/mu:

ANS: P_0 equation:

ANS: P_0:

ANS: P_1:

ANS: P_2:

ANS: P_3:

3.3 Describe how these numbers fit with your observations

Did you observe declining probabilities of higher numbers of customers when you watched the queue? List the sum of all the probabilities two ways. What is it: a) using your computations? b) using the theory? Now, in your own words, describe the difference between a uniform distribution (any number between 0 and 1 is equally likely) and the exponential distribution that you observed.

ANS: Did you see declining probabilities of higher numbers of customers in the queue display?

ANS: Sum of probabilities you computed above:

ANS: Sum of all probabilities by the formula:

ANS: Describe the difference between a uniform distribution and the exponential distribution that you observed:

4 Grammar aspects of the simulation

4.1 typedef

One of the first lines of code in the simulation code that you will be given is:

`typedef struct customer customer;`

In Kernighan and Ritchie, see Appendix A.8.9. It includes the statement: "`typedef` does not introduce new types, only synonyms for types that could be specified in another way." So the above line of code gives another way to define a customer as the given structure: you could now write `customer *z;`. In the function `*genCustomer`, there is the line: `customer *cus;`. Also see supplied file: grammar1_typedef_struct_customer_specifier. Write two different typedef statements to make two shorter synonyms for struct customer.

ANS1:

ANS2:

4.2 Time

The include statement you need is `#include <sys/time.h>`. On the server, type: `cd /usr/include` to see the possibilities for includes. The code is open source. Now type: `man timeval`. You will see the actual struct named timeval. At the end is the list of other commands to see. It includes gettimeofday and timespec. Write down the definition of struct timeval. Then go to man pages for the other two. Then write down the definition of struct timespec and struct gettimeofday. Also see the supplied files for grammar2_struct_timeval_definition and grammar3_struct_timespec_definition.

ANS: timeval:

ANS: timespec:

ANS: gettimeofday:

4.3 Draw a random number from the uniform distribution

See section 5.3 of proj4. You have to use the drand48_data setup first. Then seed the generator for drand48_r() using srand48_r(). Then run drand48_r(). Also see supplied file grammar4_struct_drand48_data_definition.

Create a program named DrawRandom to run the function listed in section 5.3 of proj4. Add a main method to call the function and print out the result. Compile and run on the server. Run it three times. List the numbers you got.

ANS1:

ANS2:

ANS3:

4.4 Use it to draw a random number from the exponential distribution

Read section 5.4 of proj4. List its four steps and briefly explain each one.

ANS1:

ANS2:

ANS3:

ANS4:

5 Proof of memorylessness

Look back at queuing slides 9 and 10. Watch the video by Lawrence Leemis on the cs444 website for Proof of Memorylessness. Write the proof here.

ANS: