




pub/sub

[Submit Assignment](#)


Due	Sunday by 11:59pm	Points	100	Submitting	a file upload	File Types	zip
------------	-------------------	---------------	-----	-------------------	---------------	-------------------	-----

in this assignment you will implement a multi-threaded pub/sub engine to track purchases.

a purchase has a timestamp associated with it (the time in milliseconds from the "beginning of time"), an amount, a string representing the place of purchase, and a description. publishers will publish purchase information using a publish function that they are given when they are started. see [pubsub.h](#)  for the declaration of `pub_init_t()` and the `publish_t()` function that the publishers will use to publish information. [simple_pub.c](#)  is an example publisher that you can test with and use to see how things work. subscribers will startup and watch for information about purchases to be published and then process them somehow. [simple_sub.c](#)  is an example subscriber.

note both publishers and subscribers can take a single argument (string) that they can use or ignore. `simple_pub` uses the argument to find how many purchases it should publish.

an implementation can provide both a publisher and subscriber like [simple.c](#)  does.

you are also provided [pubsub.c](#)  which is a simple (and incorrect) implementation of a pubsub engine. it takes a list of libraries that have publishers and subscribers along with their arguments. for example:

```
$ ./pubsub ./simple.so na ./simple_pub.so 4 ./simple_sub.so na ./simple_pub.so 1
saw 8 items
1573458850387: $1.000000 @one for first
1573458850387: $2.000000 @two for second
1573458850387: $3.000000 @three for third
1573458850387: $0.000000 @simple place for periodic purchase
1573458850487: $1.000000 @simple place for periodic purchase
1573458850591: $2.000000 @simple place for periodic purchase
1573458850696: $3.000000 @simple place for periodic purchase
1573458850797: $0.000000 @simple place for periodic purchase
```

since `simple.so` and `simple_sub.so` don't use their arguments, i'm just passing "na". we are going to run two versions of `simple_sub`: one with a count of 4 and one with a count of 1.

you will notice that when `pubsub.c` starts it loads the libraries and looks for **pub_init** and **sub_init** functions. it correctly loads them and starts them, but it doesn't start them in their own threads! you need to fix that! of course, that is going to mess up your list since you can have items being read and written and the same time, so you need to make things thread safe. you'll probably also want a list for each subscriber. your final implementation must have all the publishers and subscribers running at the same time.


compiling the code

use the following command to compile the `pubsub.c`:

```
cc -g -std=gnu11 main.c -o pubsub -ldl
```

using the following command to compile the publisher and subscriber libraries. i'm using simple.c in this example:

```
cc -g -std=c11 -shared -fPIC -o simple.so simple.c
```

DO NOT CHANGE [pubsub.h](#) ! you are only going to submit a single file: pubsub.c. so, if you change anything else for pubsub.c to work, it will not work for the graders.

rubric

publishers and subscribers run concurrently	15
everything published is received in order by the subscribers	20
correctly uses pthread threads (create and join), mutexes, and condition variables.	15
no memory leaks	10
code is threadsafe	15
no compiler warnings	5
code is easy to follow	15
only pubsub.c in submitted zipfile	5