Part2-Task1

Team Member

shengbo lou: 28530995
zhongce ji: 28551884
yucheng lu: 28411887

Compile

Go to the lab02_part2/part2_C folder

- compile: `make`
- run: `./Main`

Design

This is the Second part of the lab2, for this part we are requested to use Semaphores, so we will use C language to implements the Semaphores. We are requested to have one producer and many consumers. And the producer will produce the requests, and consumers will get the requests, and execute the requests.

- Main.c
  This is the main file to implement the whole programs. First we will have a struct which is the request struct which includes the request Id and the request length.

```
typedef struct Request{
  int ID;
  int length;
}Request;
```

Then we will have some variables:
`REQUEST_ARR`: Array of Request, this is used to store all the requests that the producer make.
`count`: int, this is used to count for the element, so we can track the requests in the array
`r_count`: int, this is used to count how many total requests we have produced.
`curr_time`: int, this will be represented as the current time.
The three int variables are very important for our implementation because we use these three variables to determine when to produce a request and when to get a request from the request list.
`mutex,empty,full`: sem_t, those three variables are the semaphores, and we will use it to lock or unlock.
Three semaphores will help us to lock the request list so producer and consumer cannot access it at the same time, and producer have to wait then the list is full and consumer has to wait when the list is empty.

`Consumer_funciton`:
This function will take one argument which is `input`, the type is `pthread_t`, and this input represents the list if threads to be created. Then we will have an infinite loop.

In the loop, first we have to request lock because we are going to get request, we use sem_wait to request lock, First we use sem_wait for full. If the full is 0, then it means it is empty so we will wait for the for requests. Then we use sem_wait for mutex which will access to butter. Then we get one of the request out of the request array. Then we get requests and record all request information, length and ID, then we decrement the count. After that we have done the get request, So we have to release the request so that other consumers can access to get the requests. Then we get the current local time and record it. After that we print out the getting requests outputs. Then we processing the requests, after we done the processing, we get the current time again, print out the complete the request output.

Producer_function:
This function will take one argument which is also input, the type is pthread_t, this input is the list if threads to be created. In the function, there is an infinite loop. In the loop, first we use sem_wait to access the buffer. First we check whether the queue is full, if the queue is not empty, then we have to create more processes and put it into the queue. So we create new request and put it into the queue and get the current_time, then print out the output of the producer. After that we can let the producer to sleep. Then we release the lock by using the sem_post.

Consumer:
This function will take two argument, one is n which means number of consumers we need to create, input is list of thread to be created. Then in the function, there is a for loop, and in the for loop, we use pthread_create to create the many threads which will take the Consumer_funciton. And then use the pthread_join for each threads.

Producer:
This function will take two argument, one is input which will be the list producer, but we one have one producer so the list size will be one. The other one is sleep_length which is the length that the producer will sleep. Then we use pthread_create to create the producer which will take the Producer_function to create the producer.

Free:
This function is used to free the requests we create.

In the main function, we first initialize the mutex, empty, full. Then we use malloc to create the requests which will have different id, then every time we will update the information about each requests. Then we call the producer function to create producer and then use consumer function to create 5 consumers. Then the producer will continue producing requests and then consumers will take the requests to process. After that the free function will free all the requests.