Parallel Programming HW3 Write Up
Due Monday February 8, 2021, 11:59 pm

Connor Osborne
A01880782

Monday February 8, 2021

1 Description

For homework 3, I implemented an MPI program that allows a process to continue working, rather than being blocked waiting for a message to be received.

The scenario is a kitchen with one cook and one or more chefs. The chefs submit orders to the cook, and the cook prepares the orders one at a time.

The cook will check to see if there are any orders. If there are orders waiting, he will collect all of them, then cook each order, one per second. If there are no orders waiting, the cook will go outside and smoke a one-second cigarette. After completing all the orders, or after smoking a cigarette, the cook will check again to see if there are orders waiting, and the process continues. If there are more than 20 orders waiting, the cook will become agitated, yell an epithet at each chef, then quit and walk out.

Each chef will check to make sure no epithets have yelled at them, then randomly take 1 to 5 seconds thinking up a new order and giving it to the cook. If the chef can see that the cook has quit, then the chef will become flustered and walk out.

The program works without recompiling for one cook and one or more chefs. If there is one cook and one chef, the program will run indefinitely. As more cooks are added, the number of orders will eventually pile up and the system will terminate.

2 Implementation

The implementation starts with creating variables for keeping track of each process and the number of processes, as well as how many orders that the chefs send in, how many orders the cook needs to do and a variable for the cook to send an epithet to the chefs if and when he quits. I also do the initial set up for the MPI processes and start a random number seed for randomizing how long it takes each chef to add an order.

```
int orders = 0;
int todo = 0;
int epithet;
int rank, size;

MPI_Status mystatus;
```

```
MPI_Init(&argc, &argv);
MPI_Comm_rank(MCW, &rank);
MPI_Comm_size(MCW, &size);

srand(rank);
```

The next step is the implementation of the cook's actions. With the cook being the process of rank 0, it begins its actions by probing to see if any orders have been sent. If something has been sent, the process enters a while loop to receive the orders and increment the total number of orders it needs to complete. Once all of the orders are accounted for the cook checks to see how many there are. If there are no orders the cook goes out for a smoke break, if there are between 1 and 20 the cook with do them one at a time while decrementing the todo variable, and if there are more than 20 orders the cook quits and sends a message to each chef with an epithet to let them know it's done.

```
if (rank == 0) {// Cook
    while (1) {
        int myFlag;
        MPI_Iprobe(MPI_ANY_SOURCE, MPI_ANY_TAG, MCW, &myFlag, &mystatus);
        while(myFlag){
            MPI_Recv(&orders, 1, MPI_INT, MPI_ANY_SOURCE, 0, MCW, MPI_STATUS_IGNORE);
            todo += orders;
            MPI_Iprobe(MPI_ANY_SOURCE, MPI_ANY_TAG, MCW, &myFlag, &mystatus);
        }
        cout << "Cook: Looks like there are " << todo << " queued up." << endl;
        if (todo == 0) {
            cout << "Cook: I'm taking a smoke break." << endl;
            sleep(1);
        }
        else if (todo > 0 && todo <= 20) {
            while (todo > 0) {
                sleep(1);
                todo -= 1;
                cout << "Cook: Finished an order, there are " << todo << " to go." << endl;
            }

            cout << "Cook: I finished the order queue. Let's see if there's anything else." <<
                endl;
        }
        else if (todo > 20) {
            //send quit message to all chefs
            cout << "Cook: That's it! I quit!!!\n*Proceeds to shout obescenities while leaving
                and knocking things over.*" << endl;
            epithet = rand() % size;
            for (int i = 1; i < size; ++i) { // send epithet to each chef before leaving
                MPI_Send(&epithet, 1, MPI_INT, i, 1, MCW);
            }
            break;
        }
    }
}
```

The last part of the implementation is the actions of the chefs. If a process is a chef it will first probe to see if the quit message has been sent by the cook. If the message was sent the chef will also quit and end it's while loop. If not the chef will proceed to randomly take 1 to 5 seconds to make an order and send it in

to the cook.

```
    else { // Chefs
        while (1) {
            int myFlag;
            MPI_Iprobe(0, MPI_ANY_TAG, MCW, &myFlag, &mystatus);
            if (myFlag) {
                if (mystatus.MPI_TAG == 1) {
                    MPI_Recv(&orders, 1, MPI_INT, 0, 1, MCW, MPI_STATUS_IGNORE);
                    cout << "Chef "<< rank <<": If he's quitting, then I'm leaving too. *Gets
                        flustered and leaves.*" << endl;
                    break;
                }
            }
            else {
                int orderMaking = rand() % 5 + 1;
                sleep(orderMaking);
                orders = 1;
                MPI_Send(&orders, 1, MPI_INT, 0, 0, MCW);
                cout << "Chef: " << rank << ": puts in an order." << endl;
            }
        }
    }
    MPI_Finalize();
    return 0;
```

3 Input and Output with 2 processes

```
afrobudha@AfroKnight:/mnt/c/Users/Sankokura/Documents/School/Parallel Program/HW3$ mpic++
    tooManyChefs.cpp
afrobudha@AfroKnight:/mnt/c/Users/Sankokura/Documents/School/Parallel Program/HW3$ mpirun -n 2
    -oversubscribe a.out


Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Chef: 1: puts in an order.
Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Cook: Looks like there are 1 queued up.
Chef: 1: puts in an order.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Cook: Looks like there are 1 queued up.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
```

```
Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Chef: 1: puts in an order.
Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Chef: 1: puts in an order.
Cook: Looks like there are 1 queued up.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 1 queued up.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Chef: 1: puts in an order.
Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Chef: 1: puts in an order.
Cook: Looks like there are 1 queued up.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 1 queued up.
Chef: 1: puts in an order.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Cook: Looks like there are 1 queued up.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Chef: 1: puts in an order.
Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Cook: Looks like there are 1 queued up.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Chef: 1: puts in an order.
Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Cook: Looks like there are 1 queued up.
Chef: 1: puts in an order.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Cook: Looks like there are 1 queued up.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 0 queued up.
```

```
Cook: Im taking a smoke break.
Chef: 1: puts in an order.
Cook: Looks like there are 0 queued up.
Cook: Im taking a smoke break.
Cook: Looks like there are 1 queued up.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 0 queued up.
Cook: Im taking a smoke break.
Chef: 1: puts in an order.
Cook: Looks like there are 0 queued up.
Cook: Im taking a smoke break.
Chef: 1: puts in an order.
Cook: Looks like there are 1 queued up.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 1 queued up.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 0 queued up.
Cook: Im taking a smoke break.
Cook: Looks like there are 0 queued up.
Cook: Im taking a smoke break.
Cook: Looks like there are 0 queued up.
Cook: Im taking a smoke break.
Chef: 1: puts in an order.
Cook: Looks like there are 0 queued up.
Cook: Im taking a smoke break.
Cook: Looks like there are 1 queued up.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 0 queued up.
Cook: Im taking a smoke break.
Cook: Looks like there are 0 queued up.
Cook: Im taking a smoke break.
Chef: 1: puts in an order.
Cook: Looks like there are 0 queued up.
Cook: Im taking a smoke break.
Cook: Looks like there are 1 queued up.
Chef: 1: puts in an order.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 0 queued up.
Cook: Im taking a smoke break.
Chef: 1: puts in an order.
Cook: Looks like there are 1 queued up.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 1 queued up.
Chef: 1: puts in an order.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 0 queued up.
Cook: Im taking a smoke break.
Cook: Looks like there are 1 queued up.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 0 queued up.
Cook: Im taking a smoke break.
```

4 Input and Output with 8 processes

```
afrobudha@AfroKnight:/mnt/c/Users/Sankokura/Documents/School/Parallel Program/HW3$ mpic++
    tooManyChefs.cpp
afrobudha@AfroKnight:/mnt/c/Users/Sankokura/Documents/School/Parallel Program/HW3$ mpirun -n 8
    -oversubscribe a.out


Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Chef: 2: puts in an order.
Chef: 5: puts in an order.
Cook: Looks like there are 0 queued up.
Cook:  Im taking a smoke break.
Chef: 5: puts in an order.
Chef: 3: puts in an order.
Chef: 4: puts in an order.
Chef: 6: puts in an order.
Cook: Looks like there are 2 queued up.
Chef: 7: puts in an order.
Chef: 6: puts in an order.
Chef: 5: puts in an order.
Chef: 3: puts in an order.
Cook: Finished an order, there are 1 to go.
Chef: 1: puts in an order.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 4 queued up.
Cook: Finished an order, there are 3 to go.
Chef: 1: puts in an order.
Chef: 2: puts in an order.
Chef: 6: puts in an order.
Chef: 5: puts in an order.
Chef: 4: puts in an order.
Cook: Finished an order, there are 2 to go.
Chef: 3: puts in an order.
Chef: 6: puts in an order.
Cook: Finished an order, there are 1 to go.
Chef: 7: puts in an order.
Chef: 5: puts in an order.
Chef: 3: puts in an order.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 5 queued up.
Chef: 1: puts in an order.
Chef: 3: puts in an order.
Cook: Finished an order, there are 4 to go.
Chef: 2: puts in an order.
Chef: 1: puts in an order.
Chef: 3: puts in an order.
Cook: Finished an order, there are 3 to go.
Chef: 4: puts in an order.
Chef: 2: puts in an order.
Chef: 5: puts in an order.
```

```
Chef: 6: puts in an order.
Cook: Finished an order, there are 2 to go.
Chef: 5: puts in an order.
Chef: 6: puts in an order.
Cook: Finished an order, there are 1 to go.
Chef: 7: puts in an order.
Chef: 4: puts in an order.
Chef: 2: puts in an order.
Chef: 3: puts in an order.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 10 queued up.
Chef: 1: puts in an order.
Chef: 6: puts in an order.
Cook: Finished an order, there are 9 to go.
Chef: 7: puts in an order.
Chef: 1: puts in an order.
Chef: 3: puts in an order.
Cook: Finished an order, there are 8 to go.
Chef: 7: puts in an order.
Cook: Finished an order, there are 7 to go.
Chef: 2: puts in an order.
Chef: 4: puts in an order.
Chef: 5: puts in an order.
Chef: 1: puts in an order.
Chef: 3: puts in an order.
Cook: Finished an order, there are 6 to go.
Chef: 6: puts in an order.
Cook: Finished an order, there are 5 to go.
Cook: Finished an order, there are 4 to go.
Chef: 7: puts in an order.
Chef: 4: puts in an order.
Chef: 1: puts in an order.
Cook: Finished an order, there are 3 to go.
Chef: 5: puts in an order.
Chef: 4: puts in an order.
Chef: 6: puts in an order.
Cook: Finished an order, there are 2 to go.
Chef: 2: puts in an order.
Chef: 7: puts in an order.
Chef: 3: puts in an order.
Cook: Finished an order, there are 1 to go.
Chef: 5: puts in an order.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 15 queued up.
Cook: Finished an order, there are 14 to go.
Chef: 1: puts in an order.
Chef: 7: puts in an order.
Chef: 4: puts in an order.
Chef: 2: puts in an order.
Chef: 5: puts in an order.
Chef: 6: puts in an order.
Cook: Finished an order, there are 13 to go.
Chef: 7: puts in an order.
Chef: 2: puts in an order.
Chef: 6: puts in an order.
Chef: 3: puts in an order.
Cook: Finished an order, there are 12 to go.
```

```
Chef: 1: puts in an order.
Cook: Finished an order, there are 11 to go.
Cook: Finished an order, there are 10 to go.
Chef: 4: puts in an order.
Chef: 5: puts in an order.
Chef: 2: puts in an order.
Cook: Finished an order, there are 9 to go.
Chef: 4: puts in an order.
Chef: 7: puts in an order.
Chef: 1: puts in an order.
Chef: 6: puts in an order.
Chef: 5: puts in an order.
Cook: Finished an order, there are 8 to go.
Chef: 7: puts in an order.
Chef: 6: puts in an order.
Chef: 3: puts in an order.
Cook: Finished an order, there are 7 to go.
Chef: 2: puts in an order.
Cook: Finished an order, there are 6 to go.
Chef: 1: puts in an order.
Chef: 7: puts in an order.
Cook: Finished an order, there are 5 to go.
Chef: 1: puts in an order.
Chef: 5: puts in an order.
Cook: Finished an order, there are 4 to go.
Chef: 4: puts in an order.
Chef: 5: puts in an order.
Chef: 7: puts in an order.
Chef: 6: puts in an order.
Chef: 3: puts in an order.
Cook: Finished an order, there are 3 to go.
Chef: 5: puts in an order.
Cook: Finished an order, there are 2 to go.
Chef: 2: puts in an order.
Chef: 7: puts in an order.
Chef: 5: puts in an order.
Cook: Finished an order, there are 1 to go.
Chef: 6: puts in an order.
Chef: 7: puts in an order.
Cook: Finished an order, there are 0 to go.
Cook: I finished the order queue. Lets see if there is anything else.
Cook: Looks like there are 22 queued up.
Cook: Thats it! I quit!!!
*Proceeds to shout obscenities while leaving and knocking things over.*
Chef: 1: puts in an order.
Chef: 5: puts in an order.
Chef: 7: puts in an order.
Chef: 4: puts in an order.
Chef: 3: puts in an order.
Chef: 6: puts in an order.
Chef: 6: puts in an order.
Chef 6: If hes quitting, then Im leaving too. *Gets flustered and leaves.*
Chef: 2: puts in an order.
Chef: 1: puts in an order.
Chef 1: If hes quitting, then Im leaving too. *Gets flustered and leaves.*
Chef: 5: puts in an order.
Chef: 7: puts in an order.
Chef 5: If hes quitting, then Im leaving too. *Gets flustered and leaves.*
Chef 7: If hes quitting, then Im leaving too. *Gets flustered and leaves.*
```

```
Chef: 3: puts in an order.
Chef 3: If hes quitting, then Im leaving too. *Gets flustered and leaves.*
Chef: 4: puts in an order.
Chef 4: If hes quitting, then Im leaving too. *Gets flustered and leaves.*
Chef: 2: puts in an order.
Chef 2: If hes quitting, then Im leaving too. *Gets flustered and leaves.*
```
