Andrew Lockwood
09/19/22
COMP 362L
LAB 04 *points 2,3, & 4*

Text document (Word or PDF) covering points 2,3 and 4 (mentioned above)

The monitor program uses the node program like a tool to reduce the temps given till a minimum threshold is reached. The monitor temp reduces at a greater rate 1/6, while dependent on the current temp of all the node. The nodes in tern reduce their temp at a rate of 1/5 but also factoring in the monitor current temp. A fork is created by monitor for each node and the node program reduces the given temp until the stable signal is received from the monitor program.

I used cmake to compile the programs into a debug folder, the object files monitor, and node are both built in the bin folder. From here, the program monitor is run with [10 2 10 10] as the argument and the program begins by printing:

```
MONTIOR: Message queue /MONITOR created.
MONTIOR: Message queue /NODE_0 created.
MONTIOR: Message queue / NODE_1 created.
```

Inspecting the code, we can see right away in monitor.c that a structure for message queue attributes is created on line 29 and followed by its initialization and when created the printf statements for the output are on lines 40 and the NODEs inside the for-loop on line 59.

This for-loop processes the nodes and does so while there are nodes to process. After creating the nodes queue and printing the 'NODE queue created' message, the loop populates the nodeData array with the new node and its temperature, including a message queue id.

Before each loop is complete, a fork for this node is created and a copy of the node is cloned and execlp is used to open the program node which is sent this node's argument for temperature where the clone morphs into the new process.

Messages between monitor and node are handled by a message structure defined in the message.h. Line 39 in node.c is the connection between forks for messages. mq_open is used to "establish a connection" between programs and is checked to ensure that the message structure is active in the other program. mq_send and mq_recieve are part of the structure and used to pass messages from each node to the monitor and from the monitor to each node.

The do-while loop is where this node program's magic happens. The loop fist sends this rounds temperature as a message. Sets previous temp equal to current temp, then the current temp equal to the product of previous temp and three, plus the product of two and the temp from monitor all divided by five. $currentTemp = (previousTemp \times 3 + 2 \times monitorMSGtemp\ ) /5$

After this, the node and the new current temp is printed. Each node runs in parallel, which can be seen in the printout. Each node fork reports and continues until the message declaring the temp stable is true.

Back in monitor, the main while-loop is doing its own calculations. On each pass of the while-loop, a for-loop is set to retrieve all new temp messages from the client forks and sum them into one variable. After this the previous integrated temp is set equal to the new integrated temp which is the updating monitor temp.

Andrew Lockwood
09/19/22
COMP 362L
LAB 04 *points 2,3, & 4*

The math for monitor.c is on line 119 where the new integrated temp is made equal to the product of two and the previous integrated temp plus the sum of the client temps, all divided by 6. $newIntegratedTemp = (2 \times previousIntegratedTemp + sumClientTemps)/6$ This new integrated temp is sent as a message to the nodes and the monitor's current temp is printed.

Following this in monitor a for-loop cycles through each node to check for stability. On the first pass a conditional trigger to identify, skip, and flag the first cycle. Then check for stability on each consecutive pass. The check for stability is done on each if the node's temperatures. "Stability" is found when the absolute value of the nodes current temp minus the nodes previous temp is less than 0.001, the Minimum change threshold defined in message.h. This means that the values of temperature for each node will get very small before triggering this flag.

When the flag is triggered, all node processes are signaled via the bool msg_send.stable message to terminate and the program prints termination messages, destroys the queues, frees node, and then exits.