System Programming 2019 Fall

Programming Assignment # 2

Due: 23:00 Tue, Nov 26, 2019

1.Problem Description

In assignment 2, you are required to implement a bidding system which will handle a sequence of competitions. The goal of this assignment is to practice how to communicate between processes through pipe and FIFO, and to understand how to use fork to create processes.

There is only one bidding system which has N players. The bidding system will distribute every eight players to a competition held by a host (H0). The host should fork itself recursively until only two players are assigned to the host.
For example, suppose there are 8 players, A, B, C, D, E, F, G, H attend this competition and assigned to one of the hosts H0. Then H0 should fork two children (H1 and H2) and assign (A, B, C, D), (E, F, G, H) to H1 and H2 respectively. H1 should fork two children again (H3 and H4) and assign (A, B), (C, D) to H3 and H4 respectively. H2 should fork two children again (H5, H6) and assign (E, F) and (G, H) to H5 and H6 respectively.
A competition will be held for 10 rounds. In each round, each player needs to announce how much he would like to spend on each item. For example, A, B, C, D, E, F, G, H decide to spend 100, 200, 300, 400, 500, 600, 700, 800 separately. A, B announce their money to H3 and H3 will find out B wins over A because 200 > 100. Then, H3 will tell the player id and spent money of B to H1. C, D announce their money to H4 and H4 will find out D wins over C because 400 > 300. Then, H4 will tell the player id and spent money of D to H1. H1 will find out D wins over B because 400 > 200 and tells the player id and spent money of D to H0. SImilar for the case in (E, F, G, H), H2 will know H wins over E, F, G and tells the player id and spent money of H to H0. Finally, H0 finds out H wins over D, and announce winner id to H1 and and H2. H1 and H2 announce the winner id to (H3, H4) and (H5, H6). Then, H3, H4, H5, H6 announce winner id to (A, B), (C, D), (E, F), (G, H) respectively.

For simplicity, we use the term **root_host** to represent the H0, **child_host** is used to represent H1, H2 and **leaf_host** is used to

represent H3, H4, H5, H6. The example above is the details about 1 round of competition. The competition will be held for 10 rounds and only one player will win the competition. The bidding system needs to schedule every 8 players to each host. That is, if there are N players, then there will be C(N, 8) competitions. However, the number of host may be more than or less than the number of competitions, and a host can only hold a competition at a time.

After a competition is finished, the host should return the scores of the players to the bidding system. The bidding system should wait until all competitions completed and rank all players according to their score.

To sum up, you are required to do the following subtasks

(1) Execute bidding system, and fork a given number of hosts. Bidding system should communicate to hosts through **FIFO.**
(2) Once the root_host receives 8 players from bidding system, it needs to fork until there are 4 leaf_hosts to handle the players. Each leaf_host will fork 2 players, and starts the competition.
(3) A leaf_host should keep the result of a competition. Once the competition is finished, it writes the result to the child_host via **pipe**, then child_host writes the result to the root_host via **pipe.** Finally, root_host will write the result to bidding system via **FIFO**. Once all competitions is finished, the bidding system shows the rankings of all players, and closes all hosts.

2. Format of Inputs & Outputs

2.1 bidding_system.c

./bidding_system [host_num] [player_num]

It requires two arguments, the number of hosts (1 ≤ host_num ≤ 10), and the number of players (8 ≤ player_num ≤ 14).

At first, the bidding system should fork and execute the number of root_hosts specified by the argument, with id from 1 to host_num. The bidding system must build FIFO to communicate before executing them.

Suppose host_num = 2, the bidding system should create *2* FIFOs

- Host.FIFO: read responses from root_host 1 and root_host2

- Host1.FIFO: write message to root_host 1
- Host2.FIFO: write message to root_host 2

The message coming from the root_host would be the rankings of players in the competition held by the root_host (described in the host.c part).

After bidding system executes a desired number of root_hosts, it should then distribute 8 players to an available root_host via FIFO. The players' id are numbered from 1 to player_num, so the messages sent to the root hosts are of the format in the following,

[player1_id] [player2_id] [player3_id] [player4_id] … [player8_id]\n

Please keep every eight player's id in ascending order.

If there is no available root_host, the bidding system should wait until one of the root_hosts return the competition result, and assign another 8 players to that root_host. There will be C(player_num, 8) competitions in total. You need to make sure that you make full use of all root_hosts and try not to let any available root_host idle.

The bidding system should keep accumulative scores of all players. The accumulative scores are initialized to 0. When a host return the competition result, the bidding system should add scores to the corresponding player accumulatively. The player in the first, second, third, fourth, fifth, sixth, seventh, eighth place gets 7, 6, 5, 4, 3, 2, 1, 0 separately. After all competitions are done, the bidding system should send -1 -1 -1 -1 -1 -1 -1 -1\n to all root_hosts, telling them that all competitions are finished so that they can exit. Then, the bidding system outputs all players' id in increasing order and their corresponding ranks, separated by a space. For example, if there are nine players and their accumulative scores are 7, 10, 3, 7, 3, 6, 7, 8, 9, the bidding system should output

1 4\n

2 1\n

3 8\n

4 4\n

5 8\n

6 7\n

7 4\n

8 3\n

9 2\n

2.2 host.c

./host [host_id] [random_key] [depth]

It requires three arguments. The id of the host, random_key and the depth of the host. The depth for root_host, child_host and leaf_host are 0, 1, 2 respectively. Random key would be an integer for a host (0 ≤ random_key ≤ 65535), and should be randomly generated unique for each root_host in the same competition. It is used to verify if a response really comes from that root_host. The host_id and random key of child_hosts and leaf_hosts should be as same as their parents.

The root_host should read from “Host[host_id].FIFO”, waiting bidding system to assign 8 players in. After knowing the players, the root_host forks 2 child_hosts, each child_hosts forks 2 leaf_hosts, each leaf_host fork 2 players and executes 2 player programs, starting a ten-round competition.

Then root_host will send 4 player_id to each child_host. The message which is sent to the first child_host (H1) is

[player1_id] [player2_id] [player3_id] [player4_id]\n

The message which is sent to the second child_host (H2) is

[player5_id] [player6_id] [player7_id] [player8_id]\n

The message which is sent to the H3 is

[player1_id] [player2_id]\n

The message which is sent to the H4 is

[player3_id] [player4_id]\n

The message which is sent to the H5 is

[player5_id] [player6_id]\n

The message which is sent to the H6 is

[player7_id] [player8_id]\n

Root_hosts write to the pipe when they are passing the player id to child hosts and they read from pipe to know the announced money of player. Child_hosts and leaf_hosts read from **standard input** and write to the pipe when they are passing the player id or winner id to their children. Child_hosts and leaf_hosts read from pipe and write to the **standard output** when they are passing the announced money of players to their parents.

The leaf_host tells all these players the winner id via pipe **between each round** to help them make their decisions. So, root_host is **NO** need to tells the winner id to the players before the first round and after tenth round. Leaf_Hosts should judge which player wins the competition and send the player_id and money of the winner to child_hosts. Child hosts will receive two player_id and money from its two children. Then, it should make a comparison and send the player_id and money of the winner to root_hosts.

The format of the message is

[player_id] [money]\n

After root_host finds out which player is the winner, it should send winner_id to child_host via pipe, child_host send the winner id which is received from root_host to leaf_host via pipe, leaf_host send the winner id to player via pipe. The format of the message is

[winner_id]\n

Remember that root_host, child_host, and leaf_host **ONLY** send player_id before the first round of competition. In the other rounds, the root_host, child_host and leaf_host should send the winner_id instead of player_id.

The root_host should continue to collect message coming from 2 child_host. The root_host accumulates each player's score, which means the number of items got in this competition. After 10 rounds, the root_hosts need to output the following to "Host.FIFO".

The format of the message is

[random_key]\n

[PlayerA_id] [PlayerA_rank]\n

[PlayerB_id] [PlayerB_rank]\n

… …

[PlayerH_id] [PlayerH_rank]\n

where the ranks are ordered from 1 to 8. If some players get the same scores, they will be ranked at the same place, and the following one will be ranked according to the number of players whose scores are higher. For example, if eight players, player_1, player_2, player_3, player_4, player_5, player_6, player_7 and player_8 get 3, 4, 3, 0, 5, 6, 7, 8 respectively. Then the host will rank player_8 the first place, and rank player_7, player_6, player_5, player_2 for second, third, fourth, fifth place respectively, and rank both player_1 and player_3 the sixth place, and rank player_4 the eighth place.

After sending out the result to bidding system, the root_host should wait until bidding system assigns another competition. However, when the root_host receives -1 -1 -1 -1 -1 -1 -1 -1 from bidding system, it indicates that all competitions are done, so the root_host should send -1 -1 -1 -1 to each child_host and exit. Child_host should send -1 -1 to each leaf_host and exit. When each leaf_host receive -1 -1, it should exit immediately.

2.3 player.c

./player [player_id]

It requires one argument. player_id should between 1 to player_num. Player should read from **standard input** and write to the **standard output.**

The player should announce their money once they are executed. At the beginning of each round (except the first round), the players should read messages from the host in certain format (which we defined in host.c). Then they should write their responses to the leaf_host in the following format,

[player_id] [money]\n

indicating the id of the player and the money that players want to pay. The above process will be repeated. The players must guarantee that it correctly gives out 10 responses. The player will exit after it gives out 10 responses, and will be executed again when competing in another competition.

Important!

To obtain a unique result, the money which is paid by a player should be as same as the (player_id*100) in each rounds. For example, suppose the player id of a player is 3, then he should announce 300 in all 10 rounds. We will grade your results according to this rules.

3.Sample Execution

$ ./bidding_system 1 8

This will run 1 root_host and 8 players. The bidding_system will create Host1.FIFO and Host.FIFO. Then, it will fork and execute:

$ ./host 0 123 0 (This is the root_host (H0))

The bidding system send to root_host(root_host read from Host1.FIFO)

1 2 3 4 5 6 7 8\n

The root_host forks two child_host.

i) ./host 0 123 1 (This is the child_host1 (H1))

ii) ./host 0 123 1 (This is the child_host2 (H2))

(i) and (ii) read player id from root_host via pipes

i) 1 2 3 4\n

ii) 5 6 7 8\n

Then, each of the child_host will fork two leaf_host.

child_host1 :

i) ./host 0 123 2 (This is the leaf_host1 (H3))

ii) ./host 0 123 2 (This is the leaf_host2 (H4))

child_host2 :

iii) ./host 0 123 2 (This is the leaf_host3 (H5))

iv) ./host 0 123 2 (This is the leaf_host4 (H6))

each leaf_host read player id from its parent child_root via pipes

i) 1 2\n

ii) 3 4\n

iii) 5 6\n

iv) 7 8\n

each leaf_host fork and executes:
leaf_host1：
$ ./player 1
$ ./player 2
leaf_host2：
$ ./player 3
$ ./player 4
leaf_host3：
$ ./player 5
$ ./player 6
leaf_host4：
$ ./player 7
$ ./player 8
Once the player is executed, then starts the competition.
Assume that player 1 pays 100, player 2 pays 200, …, player 8 pays 800. Then, each player send message to leaf_host.
For example, player 1, player 2 to leaf_host1：
1 100\n
2 200\n
leaf_host1 judges that player 2 wins and send the player_id and money to child_host1 via pipe：
2 200\n
In the same way, child_host1 judges that player 4 wins and send to root_host：
4 400\n
Finally, root_host will find out the true winner in this round (player 8) and send the player id to each child_host. child_host sends to leaf_host, and leaf_host sends to each player.
8\n
The competition will be held for 10 rounds. After 10 rounds, the root_host will send the rank of each player to bidding system via Host.FIFO.
123\n
1 2\n
2 2\n

3 2\n
4 2\n
5 2\n
6 2\n
7 2\n
8 1\n
After the bidding system get the message, because all competition are done, it will send end symbol (eight “-1”) to root_host.
-1 -1 -1 -1 -1 -1 -1 -1\n
Then the root_host will send end symbol (four “-1”) to its child_hosts, the child_host will send two “-1” to its leaf_hosts.
root_host to child_host：
-1 -1 -1 -1\n
child_host to leaf_host：
-1 -1\n
The bidding system then output the final ranks.
1 2\n
2 2\n
3 2\n
4 2\n
5 2\n
6 2\n
7 2\n
8 1\n

4.Grading

There are 6 subtasks in this assignment. You can get 7 points if you finish all of them. We will test your code on csie workstation.

1. **(1 point) Your bidding_system works fine.**

We will use your *bidding_system* as well as TA’s *host* and *player* to testify whether your *bidding_system* could successfully communicate with TA’s *host* and output correct result.

2. (1 points) Your bidding_system executes competition correctly.

You should not fork new *root_hosts* when you want to hold new

competition. That is, you can only fork *root_hosts* for *host_num* times, and assign new competition to one of them.

3. (2 points) Your host works fine.

We will use your *host* as well as TA's *bidding_system* and *player* to testify whether your *host* could successfully communicate with TA's *bidding_system* and *player*, and output correct result.

4. (1 point) Your player works fine.

We will use your player as well as TA's *bidding_system* and *host* to testify whether your *player* could successfully communicate between TA's and *host*, and output correct result.

5. (1.5 points) Completeness. If you successfully produce correct result with all your *bidding_system*, *host* and *player*.

6. (0.5 points) Produce executable files successfully. Your *Makefile* can generate *bidding_system*, *host* and *player*.

5.Notes

Remember to flush(fflush(), fsync()) whenever you write messages to pipes or FIFOs to ensure the message being correctly pass out.

6.Submission

Your assignment should be submitted to CEIBA before the deadline, or you will receive penalty. At least 5 files should be included:

1. bidding_system.c
2. host.c
3. player.c
4. Makefile (as well as other *.c files)
5. readme.txt

Since we will directly execute your Makefile, therefore you can modify the names of .c files, but Makefile should compile your source code (bidding_system.c, host.c, player.c) into three executable files named bidding_system host and player. In readme.txt, it should contain three parts.

1. Execution - Please teach us how to run your program in case we could not run your code successfully.

2. Description - Please briefly state how do you finish your program and something valuable you want to explain. What algorithm do you use in the bonus?
These files should be put inside a folder named with your student ID (in lower case) and you should compress the folder into a .tar.gz before submission. Please do not use .rar or any other file types. The commands below will do the trick. Suppose your student ID is b02902000:

$ mkdir b02902000
$ cp Makefile readme.txt *.c b02902000/
$ tar -zcvf SP_HW2_b02902000.tar.gz b02902000/
$ rm -r b02902000/

Please do NOT add executable files to the compressed file. Errors in the submission file (such as files not in a directory named with your student ID (in lower case), executable files not named correctly, and so on) may cause deduction of your credits. Submit the compressed file **SP_HW2_b02902000.tar.gz** to CEIBA.

7.Reminder

1. Plagiarism is **STRICTLY** prohibited.
2. Your credits will be deducted 5% for each day delay, but a late submission is still better than absence.
3. If you have any question, please feel free to contact us via email *ntucsiesp@gmail.com* or come to R302 during TA hours.
4. Please start your work ASAP and do not leave it until the last day!
5. **Good luck!**