Programming HW2 - Chiffon Game System

Lecture Date Empty

Due Empty

Hide 2 properties

Deadline: 11/24 (三) 23:59

1. Problem Description

Source: Instagram (@pipi.lit)

Chiffon Cake is a senior at the department of CSIE in NTU. He is a cat lover and takes great joy in manipulating the feelings of his friends. By practicing top-notch emotional blackmailing, every living being falls prey to his tactics and succumbs to whatever he demands. He really enjoyed the first three years of his university life since he had plenty of candidates to play with. However, as he turned senior, he realized that his friends have become a lot busier, and some even graduated or started internships, leaving him all alone and sad. To regain the glory of his past, Chiffon Cake decided to make some changes. He laid out a plan to play games with the new students and reward them with his tasty cakes as a means to bribe them into his emotional blackmailing target.

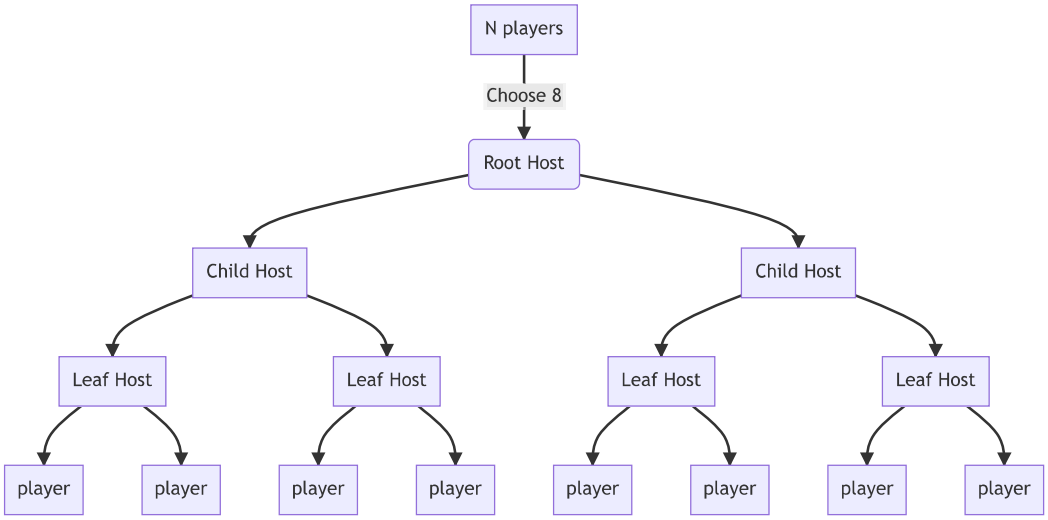
The game that Chiffon Cake wants to play is called "The Ultimate Code" (終極密碼), due to its common familiarity and high player count. Just in case you do not know the game, the rules go as the following,

1. Chiffon Cake picks his lucky lucky number ❤️ (0 - 1000).
2. All participants (8 players in our case) try to come up with a number (0 - 1000).
3. Chiffon Cake gathers the numbers and awards the participant whose number came closest to his lucky lucky number with a piece of cake. (The participant wins that round)
4. If two players guessed numbers that are the same distance to the target number, the player with the smaller `player_id` wins.

> 💡 Quick joke: Why did the school kids eat their homework?
>
> Because their teachers told them it was a piece of cake. 🎂

You, as a long-time victim of Chiffon Cake's bullying, are required to implement the game system which handles a number of games simultaneously.

ps. you gotta bring more people into the system to share the suffering QQ



Goal

1. Understand how to communicate between processes through **pipe and FIFO**.
2. Practice using fork to create processes.
3. Write a program in shell script.

There are N players using **one game system**. The game system has M hosts. Every host holds a game at a time. The game system will continuously choose 8 players and distribute them to every host., until all combinations of players are distributed. Every combination will only be distributed once, i.e. there will be $\binom{N}{8}$ games to hold. If $\binom{N}{8} > M$, there will not be enough hosts to hold all games at once. When we run out of hosts, the game system has to wait until a game ends before it can distribute another 8 players to the available host.

When a host is assigned 8 players by the game system, it will **recursively fork itself 2 times.** In other words, the root host first forks itself into two child hosts. The two child hosts then fork themselves into a total of four leaf hosts, two each. Each leaf host will be assigned 2 players.

For each game, 10 rounds of "The Ultimate Code" will be held. For each round, child hosts have to report the winners and their guesses to their parents recursively. The root host collects the winners of all 10 rounds and sums up their score. Afterward, the root host reports the players and their respective scores to the *Chiffon Game System*. The *Chiffon Game System* then transforms the sum of scores of all $\binom{N}{8}$ games into rankings and outputs the rankings. People with higher ranking means that their hearts are connected to Chiffon Cake's and deserves to be rewarded with more cake. 😍Further implementation detail will be explained later.

Example

Suppose that there are 9 players $A, B, C, D, E, F, G, H, I$ and 2 hosts H_1, H_2. We get $\binom{9}{8} = 9$ combinations of players. Say we first assign A, B, C, D, E, F, G, H to H_1 and A, B, C, D, E, F, G, I to H_2, then all the other combinations have to wait for either H_1, H_2 to finish before they can move on.

For H_1, it will fork itself into two child hosts H_{11}, H_{12}. They will be assigned A, B, C, D and E, F, G, H respectively. Same for H_2.

For H_{11}, it will further fork itself into two leaf hosts H_{111}, H_{112}. They will be assigned A, B and C, D respectively. Same for H_{12}, H_{21}, H_{22}.

In a round, H_{111} compares the guesses of A and B. It reports the winner and his guess to H_{11}. Then, H_{11} compares the winners of H_{111}, H_{112} and reports the winner and his guess to H_1. Eventually, H_1 (root host) compares the bid of the winners of H_{11}, H_{12} and gets the winner of all players.

After all 10 rounds, the root host reports the scores of all 8 players to the game system.

The *Chiffon Game System* sums up the scores of 9 games and calculates the rankings of all players.<3

> 💡 Confused? You will understand it more clearly by drawing a diagram on paper!

2. Implementation & Specs 👓

For this homework, you will have to write 3 programs. Please strictly follow the implementation guidelines as follows. Some details may be omitted on purpose below. Try your best to ensure your program runs smoothly. If you think the specs are not clear enough, feel free to ask questions on the discussion forum of NTU COOL.

chiffon.sh

- **Write this entire program in bash script <3**
  - Useful commands
    - `local`
    - `exec`
    - `read`
    - `seq`
    - `awk`
    - `mkfifo`
    - `wait`
  - Great resources for self-studying shell script
    - Chinese Mandarin: http://linux.vbird.org/linux_basic/0340bashshell-scripts.php
    - English: https://www.learnshell.org/
    - Cheatsheet: https://devhints.io/bash
- Run the program

```
bash chiffon.sh -m [n_hosts] -n [n_players] -l [lucky_number]
```

  - `n_hosts`: the number of hosts ($1 \leq M \leq 10$)
  - `n_players`: the number of players ($8 \leq N \leq 12$)
  - `lucky_number`: the number chosen by Chiffon Cake ($0 \leq L \leq 1000$)
  - Note: the command line arguments may come in any order

- Implementation (in order)
  1. Prepare **FIFO files** for all M hosts.
     - Use FIFO to communicate between the *Chiffon Game System* and hosts. Create FIFOs `fifo_0.tmp` for read and `fifo_{host_id}.tmp` for write.
     - Host ids are numbered incrementally from 1 to M.
     - Suppose $M = 2$, the *Chiffon Game System* should create 3 FIFOs.
       - `fifo_0.tmp`: read responses from host 1 and host 2.
       - `fifo_1.tmp`: write messages to host 1.
       - `fifo_2.tmp`: write messages to host 2.
  2. Generate $\binom{N}{8}$ combinations of N players and assign to hosts via FIFO. Collect scores from hosts and calculate the final result.
     - The messages sent to hosts are of the format

```
[player1_id] [player2_id] [player3_id] ... [player8_id]\n
```

     - The player ids are numbered from 1 to N.
     - Suppose players $1, 3, 5, 6, 7, 8, 9, 10$ are chosen, the message would be like

```
1 3 5 6 7 8 9 10\n
```

     - If there are no available hosts, the *Chiffon Game System* should wait until one of the hosts returns the result and assign another 8 players to that host.
       - **Do not kill the host and fork a new one.**
       - Do not leave any host idle.
     - Simply sum the scores up for all games to get the final result.
  3. After all combinations are hosted, send an ending message to all hosts forked.

```
-1 -1 -1 -1 -1 -1 -1 -1\n
```

4. Print the final rankings ordered by `player_id` (ascending) to stdout.

```
[player_id] [ranking]\n
```

5. Remove FIFO files.
6. Wait for all forked processes to exit.

host.c

- The compiled executable must be named `host`.
- Running the program

```
./host -m [host_id] -d [depth] -l [lucky_number]
```

  - It requires three arguments
    - `host_id`: the id of the host
      - `host_id` of child and leaf hosts should be the same as their parents
    - `depth`: the depth of the host
      - Starting from 0 and increments by 1 per fork
    - `lucky_number`: the number chosen by Chiffon Cake (0-1000)
    - Note: the command line arguments may come in any order

- Implementation (in order)
  1. For root host, prepare to read and write FIFO files.
  2. For hosts except for leaf hosts, fork child hosts
     - Use **pipe** to communicate between hosts. Create 2 pipes each. One for writing to child hosts. The other for reading from child host.
     - Make the forked child hosts read from **stdin** when we send `player_id` to them and write to **stdout** when they send back the winner and guess.
  3. Loop and get the player list from the parent until receiving ending message.
     a. Not leaf host: send half of the players to each child host
        a. e.g. the root host has to send

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

        to the first child host, and

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

        to the second child host. Note that the order of players has to be reserved.

     b. Is leaf host: fork *players* (refers to the program compiled from `player.c`)
        - Use **pipe** to communicate between leaf hosts and players. ***Create 1 pipe for reading from players.***
        - Make the forked *players* write to **stdout** when they send back the players and guesses.
        - Details of arguments passed to the *players* are in the next part.
     c. Iterate 10 rounds of "The Ultimate Code" game. In every round
        - For all hosts **except the root host**, read the 2 players and their guesses from child hosts or *players* via a pipe. Compare their guesses, and then send the winner and his guess to their parent host.
          - **Do not kill any of the child hosts or *players* and fork a new one**
        - For the root host, record the winner of the round and increment his

score by 10 points. In other words, the **final winner of a round gets 10 points while others do not get any points.** Sum the points up after the 10 rounds.

- ***Players* should exit after all 10 rounds are completed, fork it again next time.**

d. For root host, send the player id and their respective final scores to the Chiffon Game System.

- Format:

```
[host_id]\n [player1_id] [player1_score]\n [player2_id] [player2_score]\n ... ... [player8_id] [player8_score]\n
```

e.g. if player 1, 3, 5, 6, 7, 8, 9, 10 gets the scores of 10, 20, 10, 0, 30, 20, 10, 0 respectively, the message should be

```
[host_id]\n 1 10\n 3 20\n 5 10\n 6 0\n 7 30\n 8 20\n 9 10\n 10 0\n
```

e. Wait for forked *players* to exit and close pipes.

4. Wait for forked child hosts to exit and close pipes. For the root host, close the FIFO files.

player.c

- The compiled execution file must be named `player`
- Running the program

```
./player -n [player_id]
```

- `player_id`: the id of the player

- Implementation (in order)
  1. Iterate 10 rounds of games
     - In every round, send the player id and guess to leaf host
       - Format:

```
[player_id] [guess]\n
```

       - **The guess number must be generated using the following method to ensure consistency (Be sure to include <stdlib.h>)**

```
int guess; /* initialize random seed: */ srand ((<player_id> + <round>) * 323); /* generate guess between 1 and 1000: */ guess = rand() % 1001;
```

         - `round`: 1 to 10
         - If there exists a draw between players, the player with the **smaller player id wins**
- Note: you do not need to make all rounds start and end at the same time among hosts and players. Just keep sending messages whenever you can do so.

3. Sample Execution 💻

```
$ bash chiffon.sh -n 8 -l 323 -m 1
```

The chiffon game system here runs with 1 host, 8 players, and lucky number 323. It creates `fifo_0.tmp` and `fifo_1.tmp`. Then, it forks and executes a root host.

```
$ ./host -d 0 -l 323 -m 1
```

The chiffon game system then assigns players to the root host via `fifo_1.tmp`

```
1 2 3 4 5 6 7 8\n
```

The root host forks two child hosts.

```
$ ./host -m 1 -d 1 -l 323
```

```
$ ./host -m 1 -d 1 -l 323
```

It then assigns players to child hosts via pipes

```
1 2 3 4\n
```

```
5 6 7 8\n
```

Each child host forks two leaf hosts

```
$ ./host -m 1 -d 2 -l 323
```

```
$ ./host -m 1 -d 2 -l 323
```

It then assigns players to leaf hosts via pipes

```
1 2\n
```

```
3 4\n
```

Each leaf host forks and executes two *players*

```
$ ./player -n 1
```

```
$ ./player -n 2
```

Each player sends their guesses to their hosts (Please follow our number guessing convention above)

```
1 50\n
```

```
2 50\n
```

...

For the leaf host, it compares the guess and finds that players 1 and 2 are of the same distance to the lucky number. Thus, it sends the winner with lower player id, player 1, to its parent.

```
1 50\n
```

...

For the mid-level host that receives the above message and another similar message from another leaf host (player 3 and 4). Assume that player 4 wins, it sends the winner, player 4, to its parent

```
4 330\n
```

...

At last, the root host finds out the winner, player 8. in this round

...

After 10 rounds, the root host sends the scores to the chiffon game system via `fifo_0.tmp`.

```
1\n 1 20\n 2 0\n 3 0\n 4 30\n 5 20\n 6 0\n 7 10\n 8 20\n
```

Since all games (combinations of players) are done, the chiffon game system sends ending message to root host.

```
-1 -1 -1 -1 -1 -1 -1 -1\n
```

Then root host to its child host.

```
-1 -1 -1 -1\n
```

Then child host to leaf host.

```
-1 -1\n
```

The game system outputs the final ranking.

```
1 2\n 2 6\n 3 6\n 4 1\n 5 2\n 6 6\n 7 5\n 8 2\n
```

Note that if you follow the rules of bid, you should get

```
1 3 2 3 3 1 4 1 5 3 6 3 7 3 8 3
```

4. Notes

- Make sure to keep your fifos open, or else you may block.

```
$ exec 3<> fifo.tmp
```

5. Grading 💯

There are 5 subtasks in this assignment. You can get 8 points if you finish all of them. We will test your code on CSIE Linux workstations.

1. (*0.5 points*) *Produce executable files successfully.*

   Your Makefile generates hosts and players successfully. Running `$ make` generates all executables.

2. *(3 points) Your Chiffon Game System works smoothly.*

   We will test if your chiffon game system could successfully communicate with TA's hosts and players and outputs correct results.

   This program should be entirely in bash script.

3. *(3 points) Your host works smoothly.*

   We will test if your host could successfully communicate with TA's *Chiffon Game System* and host and outputs correct results.

4. *(1 point) Your player works smoothly.*

   We will test if your player could successfully communicate with TA's *Chiffon Game System* and hosts and outputs correct results.

5. *(0.5 point) Your own Chiffon Game System, host, and player work together smoothly*

   We will test if your own program can run all on its own

There are host and player binaries compiled by TA on the CSIE workstation in your GitHub repository. You could make use of them to write and test your code.

6. Submission

Your assignment should be submitted to GitHub before the deadline. The submission should include at least four files.

1. `Makefile`
2. `chiffon.sh`
3. `host.c`
4. `player.c`

Note that you can submit other `*.c, *.h` files as long as running `$ make` can generate all executables needed.

7. Reminders

1. Note that you have to do `fflush()` if you use `printf()` to write to the pipe.
2. Plagiarism is **STRICTLY** prohibited.
3. Your credits will be **deducted 10% for each day's delay**. Late submission is better than an absence.
4. If you have any questions, feel free to contact us via email ntucsiesp@gmail.com or come during TA hours.
5. The sample executables are compiled on the CSIE workstation. Therefore, it should only be able to run on **Linux Machines**. (Note: the sample executables are used to help you and you are not required to use it.)
6. Since everything will be judged on the CSIE workstation, make sure your final submission works smoothly on it.
7. Please start your work as soon as possible. This homework covers many important parts of the midterm. Good luck & Have fun!