CSSE2310 – Semester 2 2020

Assignment 2 (v1.1)

Your task is to write a group of three programs to participate in a naval battle tournament. One of the programs will be called 2310hub and will supervise the running of the tournament. The other two programs (2310A and 2310B) will be players (referred to as agents).

The tournament is based on the naval game, which you developed for Assignment 1. Each competition round in the tournament is a battle between two agents. The first agent to successfully sink their opponent’s ships wins their round. The hub will be responsible for running the agent processes and communicating with them via pipes (created via pipe()). From the agents’ point of view, communication will be via stdin and stdout. Agents will also produce output to stderr.

Other than the hub starting agent processes, there should not be any other parallelism used in this assignment. For example, no multi-threading nor non-blocking operations. Your programs are not permitted to start

1

any additional processes1 nor are your programs permitted to create any files during their execution.

2310hub

The hub will take the following command line arguments (in order):

• the name of the rules file

• the name of the config file

Example usage: ./2310hub rules.txt config.txt

When running agent processes, the hub must ensure that any output to stderr by agents is suppressed.

When the hub receives a SIGHUP, it should kill and reap any remaining child processes. Note: We won’t test the exit status for 2310hub when it receives SIGHUP.

Note on file handling and formatting

Input files for marking this assignment will be well-formatted, exactly according to the description and examples in this specification. This assignment is about game logic and process management and communication, notthe minutiae of parsing text files!

SPEC CLARIFICATION – The following paragraph was tweaked to make it clearer, the meaning

is unchanged In regards to message formats: most messages will consist of a certain number of comma-, colon- or spaceseparated fields. A message must contain a valid number of fields (based on its format). Each of these fields is to be processed verbatim and consist of a single token. This token can be surrounded by whitespace, but cannot contain whitespace within its body. For tokens representing/containing numeric values, the numeric component of token is considered valid provided that it can be converted successfully using standard C library functions. For example: “ GUESS D3 ” consists of two tokens (GUESS and D3) separated by whitespace, neither token contains any internal whitespace and all of the numeric components of D3 can be converted successfully using C library functions. Similarly “RULES 15, 20, 5 ” consists of four tokens (RULES, 15, 20 and 5) separated by commas, none of the tokens contain internal whitespace and all of the numeric tokens can be converted successfully.

Because your assignment and its components are auto-marked by exact text matching, you must adhere to the output formats precisely.

Gameplay

The hub uses round robin scheduling to manage competition rounds (i.e. it will circle through the rounds, prompting agents for one move each time, starting from the first competition round).

SPEC CHANGE – Some of the sequence numbers where wrong in the original version

The following describes how the hub will operate:

1. The hub will print ten asterisk characters to stdout followed by newline

\*\*\*\*\*\*\*\*\*\*

2. The hub will print the current competition round number to stdout as follows: ROUND (newline character at end) where is the round number.

3. The hub will display the boards of the two agents competing in the current round to stdout, with all ships/hits/misses revealed. See below for display details.

4. The hub will prompt the first competing agent for their move (using the YT message). The hub will continue prompting the player until a valid move is received. Note that a valid move in this case means a well-formed message with a target which is on the board which has not been targeted before.

5. After receiving a valid move from the agent, the hub replies to that agent with the OK message, and then sends either a HIT, SUNK or MISS message to both agents reflecting the result of the move.

The hub will also print information about the agent’s move to stdout as follows:

– player

guessed (newline character at end)

where

• is either HIT, MISS or SHIP SUNK,

• is the coordinate of the agent’s guess (e.g. D3), and

• is the guessing agent id (either 1 or 2).

6. If an agent’s last ship has been sunk the hub should proceed to step 8).

7. The hub will repeat steps 4 – 6 for the second competing agent in the round.

8. If game over is reached, the hub should send the DONE message to both agents in the current round and print

GAME OVER – player wins (newline character at end)

where is the id of the winning agent (1 or 2).

9. The hub should repeats steps 1-8 for each remaining competition round. Rounds that have reached the Game Over state should do steps 1 – 3, but skip the move/gameplay for that round.

The entire sequence above is repeated until all battles are completed, the hub receives SIGHUP, or some other error occurs. If the hub receives SIGHUP, it should terminate all child processes and exit.

All of the above hub output is to stdout.

Hub output example

The following shows sample output from the hub for a single round tournament.

\*\*\*\*\*\*\*\*\*\*

Round 0

ABCDEFGH

1 ..11111.

2 ……..

3 ….5…

4 ..3…..

5 2.3…..

6 2.3.44..

7 2…….

8 2…….

===

ABCDEFGH

1 1……5

2 12……

3 12..333.

4 12……

5 12…4..

6 …..4..

7 ……..

8 ……..

HIT player 1 guessed A1

SHIP SUNK player 2 guessed E3

\*\*\*\*\*\*\*\*\*\*