Project

20F-0337 20F-0320

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <stdbool.h>

#include <unistd.h>

#include <time.h>

#define ROW 5

#define COL 5

int IngredientCount = 0;

typedef struct {

pthread\_mutex\_t mutex;

char dispenser[ROW][COL];

char prevIngredient;

char selectedIngredients[ROW \* COL];

} IngredientDispenser;

typedef struct {

int black;

int red;

int blue;

int yellow;

} PotionStyle;

typedef struct {

pthread\_mutex\_t mutex;

PotionStyle\* styles;

PotionStyle completedPotions[5];

int potionRecord[5];

PotionStyle\* selectedStyles;

int completedCount;

int potionCount;

} PotionTiles;

typedef struct {

pthread\_mutex\_t mutex;

char ingredients[3];

int count;

} Flask;

typedef struct {

pthread\_mutex\_t mutex;

pthread\_cond\_t turnCV;

bool isPlayer1Turn;

IngredientDispenser dispenser;

PotionTiles tiles;

Flask extra;

} GameBoard;

typedef struct {

GameBoard\* game;

int playerId;

} PlayerThreadArgs;

void initializeDispenser(IngredientDispenser\* dispenser) {

char ingredients[4] = {'R', 'Y', 'b', 'B'};

srand(time(NULL));

for (int i = 0; i < ROW; i++) {

for (int j = 0; j < COL; j++) {

dispenser->dispenser[i][j] = ingredients[rand() % 4];

}

}

}

void\* selectIngredient(IngredientDispenser\* dispenser, int x, int y) {

pthread\_mutex\_lock(&dispenser->mutex);

dispenser->selectedIngredients[IngredientCount] = dispenser->dispenser[x][y];

dispenser->prevIngredient = dispenser->dispenser[x][y];

dispenser->dispenser[x][y] = '-';

IngredientCount++;

pthread\_mutex\_unlock(&dispenser->mutex);

return dispenser->selectedIngredients;

}

void dispenserDisplay(IngredientDispenser\* dispenser) {

for (int i = 0; i < ROW; i++) {

for (int j = 0; j < COL; j++) {

printf("%c\t", dispenser->dispenser[i][j]);

}

printf("\n");

}

}

void\* returnToDispenser(IngredientDispenser\* dispenser, char\* ingredientSelected) {

while (IngredientCount > 0) {

int tempx = rand() % ROW;

int tempy = rand() % COL;

if (dispenser->dispenser[tempx][tempy] == '-') {

dispenser->dispenser[tempx][tempy] = ingredientSelected[IngredientCount - 1];

IngredientCount--;

}

}

pthread\_exit(NULL);

}

void displayAllPotionStyles(PotionTiles\* tiles) {

printf("All 5 styles are : \n");

for (int i = 0; i < 5; i++) {

printf("Style %d : \tblack: %d\tblue: %d\tred: %d\tyellow: %d\n", i + 1, tiles->styles[i].black, tiles->styles[i].blue, tiles->styles[i].red, tiles->styles[i].yellow);

}

}

void placePotionTiles(PotionTiles\* tiles) {

pthread\_mutex\_lock(&tiles->mutex);

if (tiles->potionCount >= 1) {

printf("No more potion tiles can be placed");

pthread\_mutex\_unlock(&tiles->mutex);

return;

} else {

displayAllPotionStyles(tiles);

int choice = 0;

while (1) {

printf("Enter your choice: ");

scanf("%d", &choice);

if (choice > 0 && choice < 6) {

printf("That's a valid choice.");

tiles->potionCount++;

tiles->potionRecord[choice - 1] = 1;

pthread\_mutex\_unlock(&tiles->mutex);

return;

} else {

printf("Invalid choice");

}

}

}

}

bool completedPotions(PotionTiles\* tiles) {

bool flag = false;

for (int i = 0; i < ROW; i++) {

for (int j = 0; j < 2; j++) {

if (tiles->selectedStyles[j].black == tiles->styles[i].black &&

tiles->selectedStyles[j].blue == tiles->styles[i].blue &&

tiles->selectedStyles[j].red == tiles->styles[i].red &&

tiles->selectedStyles[j].yellow == tiles->styles[i].yellow) {

tiles->completedPotions[tiles->completedCount] = tiles->selectedStyles[j];

for (int k = j; k < tiles->potionCount - 1; k++) {

tiles->selectedStyles[k] = tiles->selectedStyles[k + 1];

}

tiles->potionRecord[i] = 0;

tiles->potionCount--;

j--;

flag = true;

}

}

}

return flag;

}

bool placeIngredients(PotionTiles\* tiles, char ingredientSelected) {

int ingredients[4];

printf("Place ingredients on previously selected potion tiles or place them in the flask.\n");

printf("Previously selected potions are:\n");

int j = 1;

for (int i = 0; i < 5; i++) {

if (tiles->potionRecord[i] == 1) {

printf("Choice: %d\n", j);

printf("Style %d:\tblack: %d\tblue: %d\tred: %d\tyellow: %d\n", i + 1, tiles->styles[i].black, tiles->styles[i].blue, tiles->styles[i].red, tiles->styles[i].yellow);

j++;

}

}

if (j == 1) {

printf("Choice 2 == Choice 1\n");

}

int temp;

while (1) {

printf("Enter your choice (1 or 2): ");

scanf("%d", &temp);

if (temp == 1 || temp == 2) {

break;

} else {

printf("Invalid choice");

}

}

bool flag = false;

int check = -1;

for (int i = 0; i < 5; i++) {

if (tiles->potionRecord[i] == 1) {

check++;

if (check == temp - 1) {

if (tiles->selectedStyles[temp - 1].black < tiles->styles[i].black) {

if (ingredientSelected == 'B') {

tiles->selectedStyles[temp - 1].black++;

printf("Ingredient is placed on potion tile\n");

flag = true;

}

}

}

}

}

for (int i = 0, j = 0; i < 5; i++) {

if (tiles->potionRecord[i] == 1) {

printf("Ingredients needed to complete Choice %d: ", j + 1);

printf("Style %d:\tblack: %d\tblue: %d\tred: %d\tyellow: %d\n",

i + 1,

tiles->styles[i].black - tiles->selectedStyles[j].black,

tiles->styles[i].blue - tiles->selectedStyles[j].blue,

tiles->styles[i].red - tiles->selectedStyles[j].red,

tiles->styles[i].yellow - tiles->selectedStyles[j].yellow);

j++;

}

}

if (!flag) {

printf("Sorry, Ingredient can't be placed on the selected potion tile\n");

}

return flag;

}

void placeInFlask(Flask\* flask, char ingredient) {

pthread\_mutex\_lock(&flask->mutex);

if (flask->count < 3) {

flask->ingredients[flask->count] = ingredient;

flask->count++;

printf("Ingredient placed in Flask.\n");

} else {

printf("Sorry! No more ingredients can be placed in Flask.\n");

}

pthread\_mutex\_unlock(&flask->mutex);

}

void displayFlask(Flask\* flask) {

for (int i = 0; i < 3; i++) {

printf("%d. %c\n", i, flask->ingredients[i]);

}

}

void\* playerAction(GameBoard\* game, int playerId) {

while (1) {

pthread\_mutex\_lock(&game->mutex);

while ((playerId == 1) == game->isPlayer1Turn) {

pthread\_cond\_wait(&game->turnCV, &game->mutex);

}

int temp;

if (game->tiles.potionCount == 1) {

if (completedPotions(&game->tiles)) {

placePotionTiles(&game->tiles);

} else {

int x, y;

dispenserDisplay(&game->dispenser);

printf("Enter (x, y): ");

scanf("%d %d", &x, &y);

char\* ingredientSelected = selectIngredient(&game->dispenser, x, y);

dispenserDisplay(&game->dispenser);

int select;

int choice;

printf("\n1. Place.\n2. Return to Dispenser.\nYou want to place ingredients or return remaining ingredients: ");

scanf("%d", &choice);

if (choice == 1) {

while (IngredientCount != 0) {

for (int i = 0; i < IngredientCount; i++) {

printf("%d. %c\n", i, ingredientSelected[i]);

}

printf("\nEnter selected ingredient (example: 1): ");

scanf("%d", &select);

int temp;

printf("\n1. Place on potion Tile.\n2. Place in flask.\nYou want to place ingredients or return remaining ingredients: ");

scanf("%d", &temp);

if (temp == 1) {

if (placeIngredients(&game->tiles, ingredientSelected[select])) {

ingredientSelected[select] = '-';

for (int i = 0; i < ROW - 1; i++) {

if (ingredientSelected[i] == '-') {

char temp = ingredientSelected[i];

ingredientSelected[i] = ingredientSelected[i + 1];

ingredientSelected[i + 1] = temp;

}

}

IngredientCount--;

}

} else if (temp == 2) {

placeInFlask(&game->extra, ingredientSelected[select]);

ingredientSelected[select] = '-';

for (int i = 0; i < ROW - 1; i++) {

if (ingredientSelected[i] == '-') {

char temp = ingredientSelected[i];

ingredientSelected[i] = ingredientSelected[i + 1];

ingredientSelected[i + 1] = temp;

}

}

IngredientCount--;

}

}

} else if (choice == 2) {

returnToDispenser(&game->dispenser, ingredientSelected);

}

}

} else {

placePotionTiles(&game->tiles);

}

game->isPlayer1Turn = !game->isPlayer1Turn;

pthread\_cond\_signal(&game->turnCV);

pthread\_mutex\_unlock(&game->mutex);

usleep(1000);

}

}

void initializePotionTiles(PotionTiles\* tiles) {

tiles->styles = (PotionStyle\*)malloc(5 \* sizeof(PotionStyle));

tiles->selectedStyles = (PotionStyle\*)malloc(5 \* sizeof(PotionStyle));

for (int i = 0; i < 5; ++i) {

tiles->styles[i].black = 1;

tiles->styles[i].red = 1;

tiles->styles[i].blue = 1;

tiles->styles[i].yellow = 1;

tiles->selectedStyles[i].black = 0;

tiles->selectedStyles[i].red = 0;

tiles->selectedStyles[i].blue = 0;

tiles->selectedStyles[i].yellow = 0;

}

}

void\* startPlayer(void\* args) {

PlayerThreadArgs\* threadArgs = (PlayerThreadArgs\*)args;

playerAction(threadArgs->game, threadArgs->playerId);

return NULL;

}

void startGame(GameBoard\* game) {

pthread\_t player1, player2;

PlayerThreadArgs\* args1 = (PlayerThreadArgs\*)malloc(sizeof(PlayerThreadArgs));

args1->game = game;

args1->playerId = 1;

PlayerThreadArgs\* args2 = (PlayerThreadArgs\*)malloc(sizeof(PlayerThreadArgs));

args2->game = game;

args2->playerId = 2;

initializePotionTiles(&game->tiles);

pthread\_create(&player1, NULL, startPlayer, args1);

pthread\_create(&player2, NULL, startPlayer, args2);

pthread\_join(player1, NULL);

pthread\_join(player2, NULL);

free(args1);

free(args2);

}

int main() {

srand(time(NULL));

GameBoard game;

initializeDispenser(&game.dispenser);

pthread\_mutex\_init(&game.mutex, NULL);

pthread\_cond\_init(&game.turnCV, NULL);

startGame(&game);

pthread\_mutex\_destroy(&game.mutex);

pthread\_cond\_destroy(&game.turnCV);

return 0;

}