#include <stdbool.h>  
#include <stdio.h>  
#include <stdlib.h>  
  
#include "../stack/stack.c"  
#include "../fifo/fifo.c"  
  
typedef struct {  
 char Name[10]; // имя запроса  
 unsigned Time; // время обслуживания  
 char P; /\* приоритет задачи: 0 — высший, 1 — средний, 2 — низший \*/  
} TInquiry;  
  
typedef struct {  
 TInquiry \*cur\_task;  
 bool has\_task;  
 char delayedP;  
} Processor;  
  
void transferTaskToProcessor(Processor \*P, Fifo \*F1, Fifo \*F2, Fifo \*F3) {  
 char priority = findHighestPriority(F1, F2, F3);  
 if (priority == 0) {  
 GetFifo(F1, P->cur\_task);  
 } else if (priority == 1) {  
 GetFifo(F2, P->cur\_task);  
 } else if (priority == 2) {  
 GetFifo(F3, P->cur\_task);  
 }  
}  
  
void shiftOnHigherPriority(Processor \*P, Stack \*S, Fifo \*F1, Fifo \*F2, Fifo \*F3) {  
 char priority = findHighestPriority(F1, F2, F3);  
 PutStack(S, P->cur\_task);  
 transferTaskToProcessor(P, F1, F2, F3);  
 P->delayedP = priority;  
}  
  
void makeTact(Generator \*G, Processor \*P, Stack \*S, Fifo \*F1, Fifo \*F2, Fifo \*F3) {

//\*\*\*//  
 } else if (findHighestPriority(F1, F2, F3) < P->cur\_task->P) {  
 printf("Found more prioritised task\n");  
 shiftOnHigherPriority(P, S, F1, F2, F3);  
 printf("Task transferred to stack: ");  
 TInquiry \*last\_delayed\_task = malloc(sizeof(TInquiry));  
 ReadStack(\*S, last\_delayed\_task);  
 printf("%s, %d, %d\n", last\_delayed\_task->Name, last\_delayed\_task->Time, last\_delayed\_task->P);  
 }  
 //\*\*\*//  
int main() {  
 Generator G;  
 Processor P;  
 Fifo F1, F2, F3;  
 Stack S;  
 InitFifo(&F1, sizeof(TInquiry), 50);  
 InitFifo(&F2, sizeof(TInquiry), 50);  
 InitFifo(&F3, sizeof(TInquiry), 50);  
 InitStack(&S, sizeof(TInquiry));  
 G.has\_task = 0;  
 G.tasks\_before = 0;  
 P.has\_task = 0;  
 P.delayedP = 2;  
  
 for (int time = 0; time < 20; time++) {  
 printf("\nTIME: %d\n", time);  
 makeTact(&G, &P, &S, &F1, &F2, &F3);  
 }  
}

Строго говоря, это не помогло (стало только хуже). Мне кажется, я интуитивно понимаю проблему, но не могу понять, как решить ее. Еще раз, помогите мне понять, может, в системе указателей есть какая-то ошибка?

1. Мы имеем структуру Inquiry

2. Мы имеем структуру Processor, одно из полей которой — указатель на экзмепляр TInquiry

3. Мы имеем структуры Stack и Fifo, Fifo реализована на массиве, Stack — на последовательном линейном списке. Массивы состоят из указателей типа void\*, которые указывают на области памяти, где хранятся настоящие элементы стека или очереди.

4. Функции PutStack(Stack \*S, void \*E) и PutFifo(Fifa \*F, void \*E) принимают как элемент пустой указатель и присваивают значение нужному элементу (вершине стека или хвосту очереди) по следующей схеме: f->Buf[f->Uk2] = E;  
5. Функции GetStack(Stack \*S, void \*E) и GetFifo(Fifa \*F, void \*E) принимают как адрес переменной, куда нужно извлечь элемент, пустой указатель и присваивают ему значение вершины стека или головы очереди по следующей схеме: memcpy(E, f->Buf[f->Uk1 ], f->SizeEl);  
6. Функция transferTaskToProcessor(Processor \*P, Fifo \*F1, Fifo \*F2, Fifo \*F3) вызывает в своем теле функцию GetFifo(F1, P->cur\_task); Вызов P->cur\_task должен вернуть указатель на задачу, которая хранится сейчас в процессоре, при этом сама задача хранится отдельно от процессора --– процессор хранит только указатель. Поэтому вызов GetFifo(F1, P->cur\_task) должен перезаписать значение по этому указателю; таким образом, изменится и задача в процессоре, потому что не изменившийся указатель будет теперь вести к другому значению.

7. Функция shiftOnHigherPriority(Processor \*P, Stack \*S, Fifo \*F1, Fifo \*F2, Fifo \*F3) вызывает в своем теле функцию PutStack(S, P->cur\_task); Вызов P->cur\_task должен вернуть указатель на задачу, которая хранится сейчас в процессоре. Поэтому вызов PutStack(S, P->cur\_task) должен приравнять значение указателя на элемент, являющийся вершиной стека, к указателю на задачу в процессоре. После этого вызывается функция transferTaskToProcessor, обновляющая задачу в самом процессоре.

8. Предполагается, что функция PutStack изменит значение указателя, по которому нужно перейти, чтобы получить элемент стека, и отныне он будет вести к той области памяти, где лежала прошлая задача процессора. А функция transferTaskToProcessor должна изменить текущую задачу процессора. Но когда мы после этих действий читаем элемент стека, оказывается, что он равен не прошлой задаче процессора, а той, что была помещена в процессор вместо прошлой. Как исправить это противоречие?