ИДЗ 2 По ОС

Бабушкин Владимир Александрович, БПИ 237

Вариант 24

условие

```
Задача о программистах. В отделе работают три программи-
ста. Каждый программист пишет свою программу и отдает ее на
проверку одному из двух оставшихся программистов, выбирая его
случайно и ожидая окончания проверки. Программист начинает
проверять чужую программу, когда его собственная уже написа-
на и передана на проверку. По завершении проверки, программист
возвращает программу с результатом (формируемым случайно по
любому из выбранных Вами законов): программа написана пра-
вильно или неправильно. Программист «спит», если отправил свою
программу и не проверяет чужую программу. Программист «про-
сыпается», когда получает заключение от другого программиста.
Если программа признана правильной, программист пишет другую
20
программу, если программа признана неправильной, программист
исправляет ее и отправляет на проверку тому же программисту,
который ее проверял. К исправлению своей программы он присту-
пает, завершив проверку чужой программы. При наличии в очере-
ди проверяемых программ и приходе заключения о неправильной
своей программы программист может выбирать любую из возмож-
ных работ. Необходимо учесть, что кто-то из программистов может
получать несколько программ на проверку.
Создать многопроцессное приложение, моделирующее ра-
боту программистов.
Каждый программист – это отдельный ПРОЦЕСС
```

Идея решения

Есть три потока, на каждый из них есть свой семафор, обозначающий количество нерешенных задач, которые предстоит сделать этому программисту (как проверок так и написания кода). А также очередь, содержащая задачи, задачи я описывал следующим видом:

```
    // -1 собственная задача
    // 0  задча программиста 0, нужно ему вернуть если ннеправильная
    // 1  задча программиста 1, нужно ему вернуть если ннеправильная
    // 2  задча программиста 2, нужно ему вернуть если ннеправильная
    // 3 задачу уже проверял программист 0, нужно снова отправить ее на
проверку
    // 4 задачу уже проверял программист 1, нужно снова отправить ее на
проверку
```

```
    // 5 задачу уже проверял программист 2, нужно снова отправить ее на
проверку
```

И так, программист ждет пока в его очереди не появится задча (sem_wait), после чегосчитывает задачу из очереди, выполняет ее, и перенаправляет другому программисту (sem_post), тот программист с вероятностью 0,7 принимает задачу, или с вероятностью 0,3 возвращает ее на переделку.

Если программист правильно выполнил задачу, то проверяющий дает ему новую, я это сделал для поддержания бесконечной симуляции работы программистов. Так, симуляция заканчивается только по воле нас людей, запускающих симуляцию.

4-5

код

В этой части я использовал именованные семафоры POSIX

В разделяемой памяти я сделал такую структуру:

```
typedef struct {
    size_t i;
    size_t j;
    size_t k;
    int queue1[50];
    int queue2[50];
    int queue3[50];
} shared_mem_t;
```

то есть, очереди ограничены длиной 50, будем считать что слишком большого завала на работе у этих программистов не случается

i, j, k это индексы последнего элемента в соответствующем массиве.

Также написал обработку SEGINT для корректной обработки ctrl+C, для этого добавил глобальную атомарную переменную, хранящую булево значение - освобождали ли уже ресурсы или нет

тестирование

поскольку моя программа по условию должна иметь элемент случайности (написание корректного кода), я решил что бессмысленно делать тесты для программ, поскольку они всегда будут выдавать разные результаты

В итоге в начале каждой программы я назначаю каждому программисту по одной задаче, и смотрю как они справляются

результат работы программы:

```
Programmer 0 is writing his own program
Programmer 1 is writing his own program
Programmer 2 is writing his own program
Programmer 1 sends his program to programmer 0 for review
Programmer 2 sends his program to programmer 1 for review
Programmer 1 is reviewing programmer 2's program
Programmer 0 sends his program to programmer 1 for review
Programmer 0 is reviewing programmer 1's program
Programmer 0 says programmer 1's program is CORRECT
Programmer 1 says programmer 2's program is CORRECT
Programmer 1 is reviewing programmer 0's program
Programmer 2 is writing his own program
Programmer 2 sends his program to programmer 1 for review
Programmer 1 says programmer 0's program is INCORRECT
Programmer 1 is writing his own program
Programmer 1 sends his program to programmer 0 for review
Programmer 1 is reviewing programmer 2's program
Programmer 0 is reviewing programmer 1's program
Programmer 0 says programmer 1's program is INCORRECT
Programmer 0 is re-reviewing programmer 1's program
Programmer 1 says programmer 2's program is CORRECT
Programmer 1 is re-reviewing programmer 0's program
Programmer 2 is writing his own program
Programmer 0 says programmer 1's program is NOW CORRECT
Programmer 1 says programmer 0's program is NOW CORRECT
Programmer 1 is writing his own program
Programmer 0 is writing his own program
Programmer 2 sends his program to programmer 1 for review
Programmer 1 sends his program to programmer 0 for review
Programmer 1 is reviewing programmer 2's program
Programmer 0 sends his program to programmer 2 for review
Programmer 0 is reviewing programmer 1's program
Programmer 2 is reviewing programmer 0's program
^C
```

6-7

код

В целом тут все то же самое, только теперь семафоры неименованные, поэтому я их храню в разделяемой памяти

```
typedef struct {
    size_t i;
    size_t j;
    size_t k;
    int queue1[50];
    int queue2[50];
    int queue3[50];
    sem_t mutex;
```

```
    sem_t task_sems[3];
} shared_mem_t;
```

мьютекс нужен для пресечения гонки при обращении к разделяемой памяти

Также изменил все функции работы с именованными семафорами на их аналоги.

пример работы программы:

```
Programmer 0 is writing his own program
Programmer 1 is writing his own program
Programmer 2 is writing his own program
Programmer 0 sends his program to programmer 1 for review
Programmer 2 sends his program to programmer 0 for review
Programmer 0 is reviewing programmer 2's program
Programmer 1 sends his program to programmer 0 for review
Programmer 1 is reviewing programmer 0's program
Programmer 0 says programmer 2's program is INCORRECT
Programmer 0 is reviewing programmer 1's program
Programmer 1 says programmer 0's program is INCORRECT
Programmer 1 is re-reviewing programmer 0's program
Programmer 1 says programmer 0's program is NOW CORRECT
Programmer 0 says programmer 1's program is INCORRECT
Programmer 0 is re-reviewing programmer 2's program
Programmer 0 says programmer 2's program is NOW CORRECT
Programmer 0 is writing his own program
Programmer 2 is writing his own program
Programmer 2 sends his program to programmer 0 for review
Programmer 0 sends his program to programmer 2 for review
Programmer 0 is re-reviewing programmer 1's program
Programmer 2 is reviewing programmer 0's program
Programmer 0 says programmer 1's program is STILL INCORRECT
Programmer 0 is reviewing programmer 2's program
Programmer 2 says programmer 0's program is CORRECT
Programmer 0 says programmer 2's program is CORRECT
Programmer 0 is re-reviewing programmer 1's program
Programmer 2 is writing his own program
Programmer 2 sends his program to programmer 1 for review
Programmer 1 is reviewing programmer 2's program
Programmer 0 says programmer 1's program is NOW CORRECT
Programmer 0 is writing his own program
^C
```

8

код

теперь, нужно запускать процессы независимо, поэтому я стал принимать аргументы у программы

```
Usage: program <programmer_id>
```

где если запустить ***program -1***, то запустится программа инициализирующая семафоры и разделяемую память

для удобства был написан bash script:

```
#!/bin/bash
gcc -o program main.c
./program -1 &


echo "use 'killall program' to stop simulation"
sleep 2
kill program


./program 0 &
./program 1 &
./program 2 &
```

Также для этой версии, все семафоры были переписаны на интерфейс SYSTEM-V

результат работы програмы:

```
[admin@babushkin05server 8]$ ./run
use 'killall program' to stop simulation
Shared memory and semaphores initialized. You can now run programmers.
Programmer 0 is writing his own program
Programmer 1 is writing his own program
Programmer 2 is writing his own program
[admin@babushkin05server 8]$ Programmer 1 sends his program to programmer
2 for review
Programmer 2 sends his program to programmer 0 for review
Programmer 2 is reviewing programmer 1's program
Programmer 0 sends his program to programmer 1 for review
Programmer 0 is reviewing programmer 2's program
Programmer 1 is reviewing programmer 0's program
Programmer 1 says programmer 0's program is CORRECT
Programmer 2 says programmer 1's program is CORRECT
Programmer 1 is writing his own program
Programmer 0 says programmer 2's program is INCORRECT
Programmer 0 is writing his own program
Programmer 0 sends his program to programmer 2 for review
Programmer 0 is re-reviewing programmer 2's program
Programmer 2 is reviewing programmer 0's program
Programmer 1 sends his program to programmer 2 for review
Programmer 0 says programmer 2's program is NOW CORRECT
Programmer 2 says programmer 0's program is CORRECT
```

```
Programmer 2 is reviewing programmer 1's program
Programmer 0 is writing his own program
Programmer 0 sends his program to programmer 2 for review
Programmer 2 says programmer 1's program is CORRECT
Programmer 2 is writing his own program
Programmer 1 is writing his own program
Programmer 2 sends his program to programmer 1 for review
Programmer 2 is reviewing programmer 0's program
Programmer 1 sends his program to programmer 2 for review
Programmer 1 is reviewing programmer 2's program
```

Для того чтобы завершить симуляцию, как написано в первой стройке, нужно в другой терминал ввести ***killall program***

9

код

В этой версии я добавил очереди сообщений стандарта SYSTEM-V, а семафоры оставил с предыдущего уровня

Поскольку, я с самого начала хранил в разделяемой памяти "очереди", придумывать как сложно привязать очереди сообщений не пришлось.

В этой версии воспользовался тем же bash скриптом,

результат работы программы:

```
[admin@babushkin05server 9]$ ./run
use 'killall program' to stop simulation
Semaphores and message queues initialized. You can now run programmers.
Programmer 0 is re-reviewing programmer 1's program
Programmer 1 is writing his own program
Programmer 2 is writing his own program
[admin@babushkin05server 9]$ Programmer 0 says programmer 1's program is
NOW CORRECT
Programmer 1 sends his program to programmer 0 for review
Programmer 1 is writing his own program
Programmer 0 is writing his own program
Programmer 2 sends his program to programmer 1 for review
Programmer 0 sends his program to programmer 2 for review
Programmer 2 is reviewing programmer 0's program
Programmer 1 sends his program to programmer 0 for review
Programmer 1 is reviewing programmer 2's program
Programmer 0 is reviewing programmer 1's program
Programmer 2 says programmer 0's program is INCORRECT
Programmer 2 is re-reviewing programmer 0's program
Programmer 0 says programmer 1's program is CORRECT
Programmer 1 says programmer 2's program is CORRECT
Programmer 1 is writing his own program
Programmer 2 says programmer 0's program is NOW CORRECT
```

```
Programmer 2 is writing his own program
Programmer 0 is reviewing programmer 1's program
Programmer 1 sends his program to programmer 0 for review
Programmer 2 sends his program to programmer 1 for review
Programmer 1 is reviewing programmer 2's program
Programmer 0 says programmer 1's program is CORRECT
Programmer 0 is writing his own program
Programmer 1 says programmer 2's program is CORRECT
Programmer 1 is writing his own program
Programmer 2 is writing his own program
Programmer 0 sends his program to programmer 2 for review
Programmer 2 sends his program to programmer 1 for review
Programmer 2 is reviewing programmer 0's program
Programmer 1 sends his program to programmer 2 for review
Programmer 1 is reviewing programmer 2's program
Programmer 1 says programmer 2's program is CORRECT
Programmer 2 says programmer 0's program is CORRECT
Programmer 2 is reviewing programmer 1's program
Programmer 0 is reviewing programmer 1's program
Programmer 0 says programmer 1's program is CORRECT
Programmer 1 is writing his own program
Programmer 2 says programmer 1's program is CORRECT
Programmer 2 is writing his own program
Programmer 1 sends his program to programmer 2 for review
Programmer 1 is writing his own program
```

10

код

Здесь все примерно то же самое, только поменялся синтаксис работы с очередями сообщений

Также использую bash скрипт

результат работы программы:

```
 ./run
use 'killall program' to stop simulation
Semaphores and message queues initialized. You can now run programmers.
Programmer 0 is writing his own program
Programmer 1 is writing his own program
Programmer 2 is writing his own program
[admin@babushkin05server 10]$ Programmer 2 sends his program to programmer
0 for review
Programmer 1 sends his program to programmer 2 for review
Programmer 2 is reviewing programmer 1's program
Programmer 0 sends his program to programmer 1 for review
Programmer 0 is reviewing programmer 2's program
Programmer 1 is reviewing programmer 0's program
Programmer 2 says programmer 1's program is INCORRECT
Programmer 2 is re-reviewing programmer 1's program
Programmer 1 says programmer 0's program is CORRECT
```

```
Programmer 0 says programmer 2's program is INCORRECT
Programmer 0 is writing his own program
Programmer 2 says programmer 1's program is STILL INCORRECT
Programmer 2 is re-reviewing programmer 1's program
Programmer 0 sends his program to programmer 2 for review
Programmer 0 is re-reviewing programmer 2's program
Programmer 2 says programmer 1's program is NOW CORRECT
Programmer 2 is reviewing programmer 0's program
Programmer 1 is writing his own program
Programmer 0 says programmer 2's program is NOW CORRECT
Programmer 1 sends his program to programmer 0 for review
Programmer 0 is reviewing programmer 1's program
Programmer 2 says programmer 0's program is CORRECT
Programmer 2 is writing his own program
Programmer 0 says programmer 1's program is CORRECT
Programmer 0 is writing his own program
Programmer 1 is writing his own program
Programmer 2 sends his program to programmer 1 for review
Programmer 0 sends his program to programmer 1 for review
Programmer 1 sends his program to programmer 2 for review
Programmer 1 is reviewing programmer 2's program
Programmer 2 is reviewing programmer 1's program
Programmer 1 says programmer 2's program is CORRECT
Programmer 1 is reviewing programmer 0's program
Programmer 2 says programmer 1's program is CORRECT
Programmer 2 is writing his own program
Programmer 2 sends his program to programmer 0 for review
Programmer 0 is reviewing programmer 2's program
Programmer 1 says programmer 0's program is INCORRECT
Programmer 1 is writing his own program
Programmer 1 sends his program to programmer 2 for review
Programmer 1 is re-reviewing programmer 0's program
Programmer 2 is reviewing programmer 1's program
Programmer 0 says programmer 2's program is CORRECT
Programmer 1 says programmer 0's program is NOW CORRECT
Programmer 0 is writing his own program
Programmer 2 says programmer 1's program is INCORRECT
Programmer 2 is writing his own program
Programmer 0 sends his program to programmer 1 for review
Programmer 1 is reviewing programmer 0's program
Programmer 2 sends his program to programmer 0 for review
Programmer 2 is re-reviewing programmer 1's program
Programmer 0 is reviewing programmer 2's program
Programmer 1 says programmer 0's program is INCORRECT
Programmer 1 is re-reviewing programmer 0's program
Programmer 2 says programmer 1's program is NOW CORRECT
Programmer 0 says programmer 2's program is CORRECT
Programmer 2 is writing his own program
Programmer 1 says programmer 0's program is NOW CORRECT
Programmer 1 is writing his own program
Programmer 0 is writing his own program
Programmer 2 sends his program to programmer 0 for review
Programmer 1 sends his program to programmer 0 for review
Programmer 0 sends his program to programmer 2 for review
```

```
Programmer 0 is reviewing programmer 2's program
Programmer 2 is reviewing programmer 0's program
Programmer 2 says programmer 0's program is CORRECT
Programmer 0 says programmer 2's program is CORRECT
Programmer 0 is reviewing programmer 1's program
Programmer 2 is writing his own program
```