Задания к работе №5 по профмодулю “Языки системного программирования”.

Все задания реализуются на языке программирования C (стандарт C99 и выше).
Реализованные приложения не должны завершаться аварийно.
Все ошибки, связанные с операциями взаимодействия со средствами межпроцессного взаимодействия и файлами, должны быть обработаны; все открытые средства межпроцессного взаимодействия и файлы должны быть закрыты инициирующим эти средства процессом.
В качестве средств межпроцессного взаимодействия разрешается использовать средства, описанные согласно стандартам System V и POSIX.

1. 1. Реализуйте приложение, в рамках которого процесс запрашивает дескриптор разделяемой памяти, записывает в эту память первые 52 прайориально простых числа (вычислить значения этих чисел необходимо самостоятельно).
   2. Реализуйте приложение, в рамках которого процесс запрашивает набор семафоров, каждый из которых необходимо заблокировать и разблокировать.

2. Некий университет решил продемонстрировать свою политкорректность, применив известную доктрину верховного суда США «Равенство порознь равенством не является» не только к цвету кожи, но и к полу. Результатом этого решения явились совместные ванные комнаты в общежитиях. Тем не менее, в поддержку исторически сложившийся традиции университет постановляет, что если в ванной комнате есть женщина, то другая женщина может туда зайти, а мужчина не может, и наоборот. На двери ванной есть индикатор, показывающий, в каком из трех состояний находится ванная комната:

   1. В ванной никого нет;

   2. В ванной только женщины;

   3. В ванной только мужчины.

   Реализуйте функции: woman_wants_to_enter, man_wants_to_enter, woman_leaves, man_leaves. При реализации функций допускается использование семафоров для синхронизации потоков. Примените реализованные функции в приложении, моделирующем работу ванной комнаты. В рамках приложения считается, что максимальное число человек в ванной комнате определяется параметром N, значение которого передается через аргументы командной строки при запуске приложения.

3. Крестьянину нужно перевезти через реку волка, козу и капусту. Но лодка такова, что в ней может поместиться только крестьянин, а с ним или один волк, или одна коза, или одна капуста. Но если оставить на берегу волка с козой, то волк съест козу, а если оставить на берегу козу с капустой, то коза съест капусту. Как перевезти свой груз крестьянину? Реализуйте клиент-серверный комплекс приложений, клиент которого принимает на вход команды от пользователя и делегирует их выполнение серверному приложению посредством сегмента разделяемой памяти, блокируемого семафорами. Продумайте возможность обработки сообщений от многих пользователей, для этого разработайте систему идентификации пользователей. В силу сложившихся обстоятельств крестьянин понимает достаточно ограниченный набор команд:

   1) take <object>; - взять в лодку заданный объект. Вместо <object> может быть написано wolf, goat или cabbage. При этом команда может быть выполнена только если в лодке есть место и требуемый объект находится на том же берегу, где и лодка;

   2) put; - выложить на берег то, что есть в лодке. При этом команда может быть выполнена только если в лодке помимо крестьянина есть ещё какой-либо объект;

   3) move; - переплыть реку на лодке. При этом команда может быть выполнена в любом случае вне зависимости от того, что именно находится в лодке.

   Клиентские приложения передают серверному приложению инструкции по одной из текстового файла (путь к этому файлу передаётся клиентскому приложению через аргументы строки), в котором лежит вся последовательность инструкций для крестьянина. Придумайте различные сценарии: которые решают данную задачу, а также с неверной последовательностью инструкций. Реализуйте программу так, чтобы были обработаны всевозможные ошибки времени выполнения.

4. Реализовать два процесса, которые вычисляют значения прайморialов i# для i <= 1000. При этом первый процесс (PROC_#2i+1) считает значения прайморialов для нечётных i, а второй (PROC_#2i) - значения прайморialов для чётных i. При этом, чтобы в обоих процессах не вычислять последовательность первых простых чисел, эта задача вынесена в отдельный процесс PRIMES, который после обнаружения детерминированным алгоритмом очередного простого числа p_i блокируется на $100p_i$ миллисекунд. Взаимодействие между PROC_#2i+1 и PROC_#2i реализовать посредством очереди сообщений, а между PROC_#2i+1, PROC_#2i и PRIMES - через разделяемую память, блокируемую семафорами.

5. Реализовать приложение, принимающее через аргументы командной строки путь к файлу с изображением формата .bmp и флаги с аргументами, описывающие операции, которые необходимо выполнить над изображением. Возможные флаги и их аргументы:
    - -blur <coeff>: размыть по Гауссу изображение с коэффициентом coeff (вещественное число больше либо равное 1);
    - -mono: преобразовать изображение в монохромное (чёрно-белое);
    - -negate: преобразовать изображение в негативное;
    - -rgb: для каждого пиксела значение канала R (red) -> назначить значением канала G (green), G -> B (blue), B -> R (через битовые операции);
    - -bounds <method>: выделить контуры изображения методом method: R - оператор Робертса; P - оператор Превитта; S - оператор Собела.

Количество флагов в командной строке не ограничено сверху (условно). Операции, описываемые флагами, выполняются друг за другом последовательно согласно их положению в командной строке; промежуточные результаты преобразования (между i и i+1 шагом и после последнего шага преобразований) необходимо сохранить в файл (имя файла генерируется псевдослучайно; файл находится в директории exe-файла). Например, для инструкции командной строки

app.exe 4ej_homjak.bmp -blur 3.0 -negate -bounds P -rgb -negate

Последовательность операций будет таковой:

1. Размыть по Гауссу с коэффициентом 3.0 изображение из файла “4ej_homjak.bmp” уровня директории exe-файла;

2. Преобразовать полученное в п. 1 изображение в негативное;

3. Выделить контуры изображения, полученного в п. 2, используя оператор Превитта;

4. Обменять значения R->G, G->B, B->R RGB-каналов изображения, полученного в п. 3;

5. Преобразовать полученное в п. 4 изображение в негативное.

Промежуточные состояния после 1, 2, 3, 4, 5 шагов сохраняются в файлы.