|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Робототехника и комплексная автоматизация

КАФЕДРА Системы автоматизированного проектирования (РК-6)

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

по дисциплине: «Разработка программных систем»

Студент Гусаров Аркадий Андреевич

Группа РК6-63Б

Тип задания Лабораторная работа №1

Вариант лабораторной работы 2

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_Гусаров А.A.**

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_Федорук В.Г.**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2022 г.*

ОГЛАВЛЕНИЕ

[Текст задания 3](#_Toc102125732)

[Описание структуры программы 3](#_Toc102125733)

[Используемые данные 3](#_Toc102125734)

[Блок-схема 4](#_Toc102125735)

[Пример работы программы 5](#_Toc102125736)

[Текст программы 5](#_Toc102125737)

Текст задания

Составить программу, решающую задачу разрешимости логического выражения, содержащего 3 переменные (т.е. задачу отыскания значений всех комбинаций 3 логических переменных, делающих выражение истинным).

Замечание. Логическое выражение реализовать в виде функции языка СИ с тремя аргументами.

Для проверки истинности логического выражения на каждой комбинации переменных организовать 8 параллельно выполняемых процессов, при этом родственные связи процессов имеют вид дерева с тремя ярусами. В качестве значений логических переменных целесообразно использовать значения, возвращаемые системным вызовом fork. В каждом процессе осуществить печать значений переменных и истинности выражения.

Описание структуры программы

Задача реализуется при помощи последовательных вызовов fork(). Каждый вызов fork порождает новый процесс и возвращает pid процесса родителю и 0 себе. Pid процесса – это некоторое ненулевое число. Применяя двойное отрицание к возвращаемому функцией fork() значению, первое отрицание сделает из этого числа ноль, а второе сделает из ноля один – значения логических переменных (переменная может быть либо 0 либо 1, так как двоичная система счисления). Затем они обрабатываются в функции вместе с введенными логическими операциями.

Используемые данные

Константная переменная n задает количество логических переменных в получаемом на вход выражении. В задании необходимо составить таблицу истинности для 3 логических переменных.

Функции void check(bool \*pid, int size, char\* buf) проверяет является ли выражение истиной. Здесь pid – массив логических переменных, size – размер этого массива, а buf – массив логических операций.

Функции bool switch\_op(bool a, bool b, char op) на вход получает 2 логические переменные и логический оператор и возвращает результат применения оператора к переменным.

Блок-схема

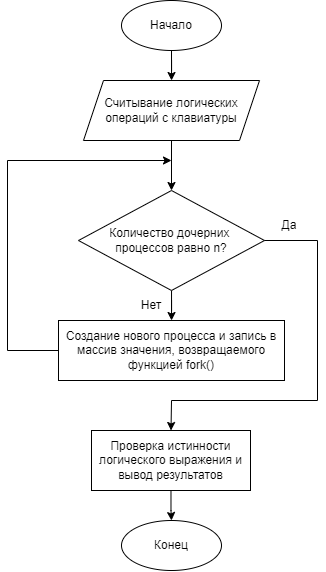


Рис. 1. Блок-схема реализованного процесса

Пример работы программы

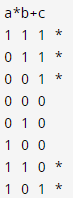
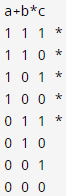
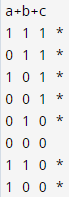
  

Рис. 2. Примеры работы программы для трёх различных входных значений a\*b+c, a+b\*c и a+b+c

Текст программы

|  |  |
| --- | --- |
|  | #include <unistd.h>  #include <stdio.h> |
|  | #include <stdlib.h> |
|  | #include <stdbool.h> |
|  |  |
|  | bool switch\_op(bool a, bool b, char op) |
|  | { |
|  | switch (op) |
|  | { |
|  | case '+': |
|  | return a || b; |
|  | case '\*': |
|  | return a && b; |
|  | } |
|  | } |
|  |  |
|  | void check(bool \*pid, int size, char \*buf) |
|  | { |
|  | bool result; // Результат сравнения |
|  | char chars[3]; |
|  | char \*smb1 = "+"; |
|  | char \*smb2 = "\*"; |
|  |  |
|  | if ((buf[0] == \*smb1) & (buf[1] == \*smb2)) |
|  | { |
|  | result = switch\_op(pid[1], pid[2], buf[1]); |
|  | result = switch\_op(result, pid[0], buf[0]); |
|  | } |
|  | else |
|  | { |
|  | result = switch\_op(pid[0], pid[1], buf[0]); |
|  | result = switch\_op(result, pid[2], buf[1]); |
|  | } |
|  |  |
|  | // Отметка, что выражение истино |
|  | if (result) |
|  | printf("\*"); |
|  | printf("\n"); |
|  | } |
|  |  |
|  | int main() |
|  | { |
|  | const int n = 3; // Количество логический переменных |
|  | bool pid[n]; // Массив логических переменных |
|  | char buf[n]; // Массив логических операций |
|  | int idx = 0; // Текущий индекс в buf |
|  |  |
|  | for (int i = 0; i < n \* 2 - 1; i++) |
|  | { |
|  | char c = getchar(); |
|  |  |
|  | if (i % 2 == 1) |
|  | { |
|  | buf[idx] = c; |
|  | idx++; |
|  | } |
|  | } |
|  |  |
|  | buf[idx] = '\0'; |
|  |  |
|  | // Генерация последовательности из 0 и 1 и вывод на экран |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | pid[i] = !!fork(); |
|  | printf("%d ", pid[i]); |
|  | } |
|  |  |
|  | // Проверка истинности выражения |
|  | check(pid, n, buf); |
|  | } |