

Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Операционные системы и системное программирование

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

по теме

ВЗАИМОДЕЙСТВИЕ И СИНХРОНИЗАЦИЯ ПРОЦЕССОВ

БГУИР КР 1-40 02 01 001 ЛР

Выполнил: студент группы 250541,
Бобрик В. Ю.

Проверил: ст.преподаватель каф. ЭВМ,
Поденок Л. П.

Минск 2025

СОДЕРЖАНИЕ

1 Условия лабораторной работы.....	3
2 Описание алгоритмов и решений.....	4
3 Функциональная структура проекта.....	5
4 Порядок сборки и тестирования.....	6
5 Методика и результаты тестирования.....	7

1 Условия лабораторной работы

Синхронизация процессов с помощью сигналов и обработка сигналов таймера.

Задание

Управление дочерними процессами и упорядочение вывода в `stdout` от них, используя сигналы `SIGUSR1` и `SIGUSR2`.

Действия родительского процесса

По нажатию клавиши «+» родительский процесс (P) порождает дочерний процесс (C_k) и сообщает об этом.

По нажатию клавиши «-» P удаляет последний порожденный C_k, сообщает об этом и о количестве оставшихся.

При вводе символа «l» выводится перечень родительских и дочерних процессов.

При вводе символа «k» P удаляет все C_k и сообщает об этом.

По нажатию клавиши «q» P удаляет все C_k, сообщает об этом и завершается.

Действия дочернего процесса

Дочерний процесс во внешнем цикле заводит будильник (`nanosleep(2)`) и входит в вечный цикл, в котором заполняет структуру, содержащую пару переменных типа `int`, значениями {0,0} и {1,1} в режиме чередования. Поскольку заполнение не атомарно, в момент срабатывания будильника в структуре может оказаться любая возможная комбинация из 0 и 1.

При получении сигнала от будильника проверяет содержимое структуры, собирает статистику и повторяет тело внешнего цикла.

Через заданное количество повторений внешнего цикла (например, через 101) дочерний процесс выводит свои PPID, PID и 4 числа — количество разных пар, зарегистрированных в момент получения сигнала от будильника.

Вывод осуществляется в одну строку.

Следует подобрать интервал времени ожидания и количество повторений внешнего цикла, чтобы статистика была значимой.

Сообщения выводятся в `stdout`.

Сообщения процессов должны содержать идентифицирующие их данные, чтобы можно было фильтровать вывод утилитой `grep`.

2 Описание алгоритмов и решений

Проект разделен на два основных модуля — один для родительского процесса, другой для дочерних процессов. Заголовочные файлы расположены в каталоге `include/`, а исходные коды — в `src/`.

Дочерний процесс выполняет следующие шаги:

Шаг 1. Внешний цикл начинается с вызова `nanosleep()`, который устанавливает задержку.

Шаг 2. После завершения задержки генерируются случайные значения для двух переменных `x` и `y` (значения 0 или 1).

Шаг 3. С помощью вызова `raise(SIGUSR1)` происходит сработка будильника.

Шаг 4. Обработанная функция вычисляет индекс комбинации (с использованием выражения $(x \ll 1) | y$) и обновляет соответствующий счетчик статистики.

Шаг 5. После прохождения определенного количества циклов дочерний процесс должен вывести накопленную статистику. Дочерний процесс переходит в режим ожидания сигнала `SIGUSR2`.

Шаг 6. При получении сигнала `SIGUSR2` обработчик устанавливает специальный флаг, разрешающий вывод.

Шаг 7. Процесс выводит строку, содержащую `PPID`, `PID` и счетчики для каждой комбинации.

Шаг 8. Сброс флага и продолжение работы.

Родительский процесс отвечает за управление жизненным циклом дочерних процессов и синхронизацию вывода.

При получении команды '+' родительский процесс выполняет `fork()`, после чего в дочернем процессе запускается соответствующее приложение с помощью `exec()`. Родитель сохраняет PID нового дочернего процесса и выводит сообщение о создании.

Команда '-' приводит к завершению последнего порождённого дочернего процесса: родительский процесс отправляет ему сигнал `SIGTERM` и ждёт завершения через `waitpid()`, после чего выводит сообщение об оставшихся процессах.

Команда 'k' осуществляет массовое завершение всех дочерних процессов, последовательно отправляя всем `SIGTERM` и подтверждая завершение каждого процесса.

Команда 'l' выводит перечень всех активных процессов.

Дополнительно введена команда 'o', при выполнении которой родительский процесс по очереди отправляет сигнал `SIGUSR2` всем дочерним процессам.

По команде 'q' родительский процесс завершает все дочерние процессы, выводит соответствующее сообщение и завершает свою работу.

3 Функциональная структура проекта

Проект разделён на два основных компонента, каждый из которых отвечает за свою функциональность и взаимодействует посредством сигналов и стандартного ввода/вывода.

Родительский процесс (P) отвечает за управление дочерними процессами и синхронизацию общего вывода.

`add_child()` – функция создания нового дочернего процесса (+).

`remove_last_child()` – функция завершения последнего созданного дочернего процесса (–).

`list_processes()` – функция вывода списка активных процессов (l);

`kill_all_children()` – функция завершения всех дочерних процессов (k);

`handle_input()` – обработчик ввода команд.

Дочерние процессы (C_k) осуществляют накопление статистических данных путём периодического обновления структуры, содержащей два поля типа `int`, и выполнение упорядоченного вывода этой статистики.

Используется структура `Data`, содержащая два целочисленных поля `x` и `y`. Эти поля обновляются случайным образом, и их значения преобразуются в индекс для массива счётчиков.

```
typedef struct {
```

```
    int x;
```

```
    int y;
```

```
} data;
```

`sigusr1_handler()` – обработчик сигнала `SIGUSR1`.

`sigusr2_handler()` – обработчик сигнала `SIGUSR2`.

`process_data()` – функция циклического изменения данных в структуре.

`print_statistics()` – функция вывода статистики по процессу.

4 Порядок сборки и тестирования

Для сборки используется `Makefile`, содержащий следующие ключевые элементы:

- Исходные файлы находятся в каталоге `./src`.

- Заголовочные файлы расположены в каталоге `./include`.

- В зависимости от типа сборки создается каталог `./out/debug` для debug-версии или `./out/release` для release-версии.

- Переменная `BUILD` задается как `debug` по умолчанию. При запуске `make release` устанавливается оптимизированная сборка.

- Флаги компиляции задаются через переменные `COMMON_CFLAGS` (определения, предупреждения, стандарт C11) и дополняются флагами отладки (`-g -O0`) или оптимизации (`-O2`).

Для сборки в debug-режиме необходимо выполнить в терминале:
`make`

Это соберет программу в `./out/debug/dirwalk`.

Для сборки в release-режиме необходимо выполнить в терминале:
`make release`

Это соберет программу в `./out/release/dirwalk`.

Для очистки сборки необходимо выполнить в терминале:
`make clean`

Для проверки работы программы выполнить:
`./out/release/parent`

Тестирование производится вручную согласно предлагаемым сообщениям.

5 Методика и результаты тестирования

Пошагово проверяется работа команд ввода.

Проверка команды создания дочернего процесса (+): ввести символ `+`.
Ожидаемый результат: родительский процесс выводит сообщение о создании дочернего процесса.

`+`

Создан дочерний процесс с PID 2038.

Проверка команды отображения перечня процессов (l): ввести символ

l. Ожидаемый результат: выводится список с PID родительского процесса и всех активных дочерних процессов.

l

Родительский процесс PID 2023.

Дочерний процесс PID 2038.

Проверка удаления последнего дочернего процесса (-): после создания нескольких дочерних процессов ввести символ -. Ожидаемый результат: вывод сообщения о завершении последнего дочернего процесса и информации о количестве оставшихся.

+

Создан дочерний процесс с PID 2041.

+

Создан дочерний процесс с PID 2042.

+

Создан дочерний процесс с PID 2043.

-

Завершен дочерний процесс с PID 2043. Осталось процессов: 3.

Проверка массового завершения (k) и полного завершения (q): ввести символ k, чтобы убедиться, что все дочерние процессы завершаются с соответствующими сообщениями. По нажатию q все процессы и родительский должны завершиться.

k

Завершен дочерний процесс с PID 3757.

Завершен дочерний процесс с PID 3758.

Завершен дочерний процесс с PID 3759.

+

Создан дочерний процесс с PID 3766.

+

Создан дочерний процесс с PID 3767.

q

Завершен дочерний процесс с PID 3766.

Завершен дочерний процесс с PID 3767.

Родительский процесс завершен.