Министерство образования Республики Беларусь
Учреждение Образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра электронных вычислительных машин

Лабораторная работа № 3
«Взаимодействие и синхронизация процессов»

Проверил: Поденок Л.П.
Выполнил: ст. гр. 350501
Кевра Н.А.

Минск 2025

1 УСЛОВИЕ 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 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 дочерний процесс выводит свои PPID, PID и 4 числа — количество разных пар, зарегистрированных в момент получения сигнала от будильника.

2 ОПИСАНИЕ АЛГОРИТМОВ И РЕШЕНИЙ

2.1 Общая структура программы

Программа состоит из двух частей: родительский процесс (parent) и дочерний процесс (child).

Родительский процесс управляет созданием, удалением и взаимодействием с дочерними процессами. Он обрабатывает команды пользователя, такие как создание нового дочернего процесса, удаление процессов, остановка и возобновление их работы. Также он отслеживает состояние дочерних процессов и их вывод.

Дочерний процесс выполняет цикл, в котором обновляет статистику (количество пар 00, 01, 10, 11). Он отправляет статистику родительскому процессу через сигналы и ожидает разрешения от родительского процесса для вывода данных.

2.2 Алгоритм работы родительского процесса

Инициализация:

Выделяется память для хранения информации о дочерних процессах (массив child_processes).

Настраиваются обработчики сигналов (SIGUSR1, SIGUSR2, SIGCHLD).

Основной цикл:

Программа ожидает ввода пользователя.

В зависимости от введённой команды выполняются соответствующие действия:

«+»: Создаётся новый дочерний процесс с помощью fork() и execl(). Информация о процессе добавляется в массив child_processes.

«-»: Удаляется последний созданный дочерний процесс. Родительский процесс отправляет сигнал SIGTERM дочернему процессу и удаляет его из массива.

«l»: Выводится информация о всех дочерних процессах (их имена и PID).

«k»: Удаляются все дочерние процессы. Родительский процесс отправляет сигнал SIGTERM каждому дочернему процессу и очищает массив child_processes.

«s<num>»: Останавливает вывод данных для дочернего процесса с указанным индексом. Родительский процесс отправляет сигнал SIGUSR1 дочернему процессу.

«g<num>»: Возобновляет вывод данных для дочернего процесса с указанным индексом. Родительский процесс отправляет сигнал SIGUSR2 дочернему процессу.

«q»: Завершает программу, удаляя все дочерние процессы и освобождая ресурсы.

Обработка сигналов:

SIGUSR1: Используется для запрета вывода данных дочернего процесса. Родительский процесс отправляет этот сигнал дочернему процессу, чтобы остановить вывод.

SIGUSR2: Используется для разрешения вывода данных дочернего процесса. Родительский процесс отправляет этот сигнал дочернему процессу, чтобы возобновить вывод.

SIGCHLD: Используется для обработки завершения дочерних процессов. Родительский процесс удаляет завершённые процессы из массива child_processes.

Управление состоянием дочерних процессов:

Родительский процесс отслеживает состояние каждого дочернего процесса (запущен или остановлен) и управляет их выводом данных.

2.3 Алгоритм работы дочернего процесса

Инициализация:

Настраиваются обработчики сигналов (SIGUSR1, SIGUSR2, SIGALRM).

Устанавливается случайный таймер для обновления статистики.

Основной цикл:

В каждом цикле обновляется статистика (количество пар 00, 01, 10, 11) с помощью функции update_stats().

Если прошло более 5 секунд и вывод разрешён (can_print == true), дочерний процесс отправляет статистику родительскому процессу через сигнал SIGUSR1.

Дочерний процесс ожидает разрешения от родительского процесса для вывода данных.

Если вывод разрешён, дочерний процесс выводит статистику и отправляет сигнал SIGUSR2 родительскому процессу, сообщая о завершении вывода.

Обработка сигналов:

SIGUSR1: Запрещает вывод данных. Устанавливает флаг can_print = false.
SIGUSR2: Разрешает вывод данных. Устанавливает флаг can_print = true.
SIGALRM: Обновляет статистику и устанавливает новый таймер.

2.4 Взаимодействие между процессами

Создание дочернего процесса:

Родительский процесс создаёт дочерний процесс с помощью fork() и execl().

Информация о новом дочернем процессе (PID и имя) добавляется в массив child_processes.

Управление выводом данных:

Родительский процесс отправляет сигналы SIGUSR1 и SIGUSR2 дочерним процессам для остановки и возобновления вывода данных.

Дочерний процесс обрабатывает эти сигналы и изменяет своё состояние (флаг can_print).

Передача статистики:

Дочерний процесс отправляет статистику родительскому процессу через сигнал SIGUSR1.

Родительский процесс обрабатывает сигнал и выводит статистику, если вывод разрешён.

Завершение работы:

При завершении программы родительский процесс удаляет все дочерние процессы, отправляя им сигнал SIGTERM, и освобождает ресурсы.

3 ФУНКЦИОНАЛЬНАЯ СТРУКТУРА ПРОЕКТА

Родительский процесс.

Родительский процесс отвечает за управление дочерними процессами, обработку команд пользователя и взаимодействие с дочерними процессами через сигналы. Основные функции родительского процесса:

Инициализация:

init_signals(): Настраивает обработчики сигналов (SIGUSR1, SIGUSR2, SIGCHLD).

Выделение памяти: Память для хранения информации о дочерних процессах выделяется при запуске программы.

Управление дочерними процессами:

create_child(): Создаёт новый дочерний процесс с помощью fork() и execl(). Информация о процессе добавляется в массив child_processes.

delete_last_child(): Удаляет последний созданный дочерний процесс.

Родительский процесс отправляет сигнал SIGTERM дочернему процессу и удаляет его из массива.

delete_all_children(): Удаляет все дочерние процессы. Родительский процесс отправляет сигнал SIGTERM каждому дочернему процессу и очищает массив child_processes.

Обработка команд пользователя:

print_menu(): Выводит меню доступных команд.

Основной цикл: Ожидает ввода пользователя и выполняет соответствующие действия в зависимости от команды.

Управление состоянием дочерних процессов:

start_child(int index): Возобновляет вывод данных для дочернего процесса с указанным индексом. Родительский процесс отправляет сигнал SIGUSR2 дочернему процессу.

stop_child(int index): Останавливает вывод данных для дочернего процесса с указанным индексом. Родительский процесс отправляет сигнал SIGUSR1 дочернему процессу.

Завершение программы:

cleanup_and_exit(): Удаляет все дочерние процессы, освобождает память и завершает работу программы.

Вспомогательные функции:

wait_for_children(): Ожидает завершения всех дочерних процессов.

handle_signal(): Обрабатывает сигналы от дочерних процессов (SIGUSR1, SIGUSR2).

handle_child_exit(): Обрабатывает завершение дочерних процессов (сигнал SIGCHLD).

Дочерний процесс

Дочерний процесс выполняет задачи, связанные с обновлением статистики (количество пар 00, 01, 10, 11), и взаимодействует с родительским процессом через сигналы. Основные функции дочернего процесса:

Инициализация:

init_signals_handling(): Настраивает обработчики сигналов (SIGUSR1, SIGUSR2, SIGALRM).

Основной цикл:

update_stats(): Обновляет статистику (количество пар 00, 01, 10, 11).

alarm_signal_handler(): Обрабатывает сигнал SIGALRM, обновляет статистику и устанавливает новый таймер.

user_signal_handler(): Обрабатывает сигналы SIGUSR1 и SIGUSR2, управляя разрешением на вывод данных.

Взаимодействие с родительским процессом:

Отправляет статистику родительскому процессу через сигнал SIGUSR1.

Ожидает разрешения от родительского процесса для вывода данных.

Отправляет сигнал SIGUSR2 родительскому процессу после завершения вывода.

4 ПОРЯДОК СБОРКИ И ЗАПУСКА

1. Перейти в каталог проекта.
   cd 'Кевра Н.А./lab03'
2. Собрать проект с помощью make.
   make
3. Перейти в каталог cd /build/debug
4. Запустить программу.
./parent

5 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

```
Options:
+ - Create new child
- - Delete last child
l - List all children
k - Delete all children
s<num> - Stop child at index <num>
g<num> - Start child at index <num>
q - Quit
m - Show this menu
> +
Created child C_00 with PID 14020
> +
Created child C_01 with PID 14022
> +
Created child C_02 with PID 14025
> g0
Started child C_00 with PID 14020
> Parent: Received SIGUSR1 from child 14020
> ----------------------------------------
ppid - 14017        pid  - 14020        00   -     4; 01   -     2; 10   -     2; 11   -
4
Parent: Child 14020 has finished output
> g1
Started child C_01 with PID 14022
> Parent: Received SIGUSR1 from child 14022
> ----------------------------------------
ppid - 14017        pid  - 14022        00   -     4; 01   -     2; 10   -     2; 11   -
4
Parent: Child 14022 has finished output
> gParent: Received SIGUSR1 from child 14020
> ----------------------------------------
ppid - 14017        pid  - 14020        00   -     5; 01   -     3; 10   -     3; 11   -
6
Parent: Child 14020 has finished output
> g2
Invalid index
```

> Parent: Received SIGUSR1 from child 14022
> --
ppid - 14017 pid - 14022 00 - 5; 01 - 3; 10 - 3; 11 - 6
Parent: Child 14022 has finished output
> Parent: Received SIGUSR1 from child 14020
> --
ppid - 14017 pid - 14020 00 - 6; 01 - 4; 10 - 4; 11 - 8
Parent: Child 14020 has finished output
> Parent: Received SIGUSR1 from child 14022
> --
ppid - 14017 pid - 14022 00 - 6; 01 - 4; 10 - 4; 11 - 8
Parent: Child 14022 has finished output
> Parent: Received SIGUSR1 from child 14020
> --
ppid - 14017 pid - 14020 00 - 7; 01 - 5; 10 - 5; 11 - 10
Parent: Child 14020 has finished output
> Parent: Received SIGUSR1 from child 14022
> --
ppid - 14017 pid - 14022 00 - 7; 01 - 5; 10 - 5; 11 - 10
Parent: Child 14022 has finished output
> Parent: Received SIGUSR1 from child 14020
> --
ppid - 14017 pid - 14020 00 - 8; 01 - 6; 10 - 6; 11 - 12
Parent: Child 14020 has finished output
> Parent: Received SIGUSR1 from child 14022
> --
ppid - 14017 pid - 14022 00 - 8; 01 - 6; 10 - 6; 11 - 12
Parent: Child 14022 has finished output
> Parent: Received SIGUSR1 from child 14020
> --

```
ppid - 14017    pid  - 14020    00  -  9; 01  -  7; 10  -  7; 11  -
14
Parent: Child 14020 has finished output
> Parent: Received SIGUSR1 from child 14022
> ----------------------------------------
ppid - 14017    pid  - 14022    00  -  9; 01  -  7; 10  -  7; 11  -
14
Parent: Child 14022 has finished output
> Parent: Received SIGUSR1 from child 14020
> ----------------------------------------
ppid - 14017    pid  - 14020    00  -  10; 01  -  8; 10  -  8; 11  -
16
Parent: Child 14020 has finished output
> Parent: Received SIGUSR1 from child 14022
> ----------------------------------------
ppid - 14017    pid  - 14022    00  -  10; 01  -  8; 10  -  8; 11  -
16
Parent: Child 14022 has finished output
> Parent: Received SIGUSR1 from child 14020
> ----------------------------------------
ppid - 14017    pid  - 14020    00  -  11; 01  -  9; 10  -  9; 11  -
18
Parent: Child 14020 has finished output
> Parent: Received SIGUSR1 from child 14022
> ----------------------------------------
ppid - 14017    pid  - 14022    00  -  11; 01  -  9; 10  -  9; 11  -
18
Parent: Child 14022 has finished output
> qParent: Received SIGUSR1 from child 14020
> ----------------------------------------
ppid - 14017    pid  - 14020    00  -  12; 01  -  10; 10  -  10; 11  -
20
Parent: Child 14020 has finished output
> q
Deleted child C_02 with PID 14025
Deleted child C_01 with PID 14022
Deleted child C_00 with PID 14020
Exiting...
```