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 №3 на тему
“Взаимодействие и синхронизация процессов”

Выполнил:
студ. гр. 350501 Везенков М. Ю.

Проверил: ст.пр. Поденок Л. П.

Минск 2025

СОДЕРЖАНИЕ

1 ЦЕЛЬ 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`.

Требования к сборке аналогичны требованиям из лабораторной № 2.

2 ВЫПОЛНЕНИЕ ЛАБОРАТОРНОЙ РАБОТЫ

2.1 Описание программы

Программа включает два основных компонента: родительский процесс (parent.c) и дочерний процесс (child.c), разработанные для демонстрации взаимодействия процессов в UNIX-подобных системах с использованием сигналов и управления процессами.

Родительский процесс отвечает за создание, завершение и мониторинг дочерних процессов на основе ввода с клавиатуры.

Дочерний процесс выполняет неатомарные обновления структуры данных, прерываемые сигналами таймера, и собирает статистику.

Родительский процесс запускается без аргументов командной строки и работает в режиме "сырого" терминала, который настраивается функциями tcsetattr() и tcgetattr(). В этом режиме программа считывает одиночные символы с клавиатуры без буферизации и интерпретирует их как команды: '+' создаёт новый дочерний процесс, '-' завершает последний созданный дочерний процесс, 'l' выводит список текущих дочерних процессов, 'k' завершает все дочерние процессы, '1' отправляет сигнал SIGUSR1 для включения вывода статистики дочерними процессами, '2' отправляет сигнал SIGUSR2 для отключения вывода статистики, а 'q' завершает программу.

Путь к исполняемому файлу дочернего процесса определяется через переменную окружения CHILD_PATH, которая должна указывать на директорию, содержащую исполняемый файл child. Родительский процесс управляет динамическим массивом PID дочерних процессов, расширяя его при необходимости с помощью realloc() и очищая при завершении или удалении процессов.

Дочерний процесс выполняет циклическое неатомарное обновление структуры pair_t, переключая её между состояниями {0,0} и {1,1}.

Обновления прерываются сигналом SIGALRM, который генерируется таймером setitimer() с интервалом 500 микросекунд. Обработчик сигнала SIGALRM фиксирует текущее состояние структуры, которое может быть {0,0}, {0,1}, {1,0} или {1,1}, и увеличивает соответствующий счётчик. После выполнения 10001 итераций, определённых константой NUM_REPETITIONS, дочерний процесс выводит статистику в стандартный поток вывода, если это не запрещено сигналом SIGUSR2, устанавливающим флаг g_output_enabled в 0. Сигнал SIGUSR1 включает вывод статистики, устанавливая g_output_enabled в 1. Для обеспечения безопасности работы с глобальными переменными в обработчиках сигналов используются типы volatile и sig_atomic_t. Программа завершает выполнение при вводе команды 'q', получении сигналов SIGINT, SIGTERM или SIGQUIT, либо при обнаружении EOF на стандартном вводе. При завершении родительский процесс отправляет

сигнал SIGKILL всем дочерним процессам, освобождает выделенную память и восстанавливает исходные настройки терминала.

2.2 Описание основных функций

Родительский процесс (parent.c)

Функция main() является точкой входа родительского процесса. Она инициализирует глобальные переменные через вызов initialize_globals(), проверяет наличие переменной окружения CHILD_PATH и доступность исполняемого файла дочернего процесса. Затем функция настраивает "сырой" режим терминала с помощью enable_raw_mode(), регистрирует обработчики сигналов через register_signal_handlers() и выделяет начальную память для массива PID дочерних процессов.

В основном цикле main() считывает команды с клавиатуры и вызывает соответствующие функции, такие как spawn_child(), kill_last_child(), list_children(), kill_all_children() или signal_all_children(). При завершении программа возвращает EXIT_SUCCESS в случае успеха или EXIT_FAILURE при возникновении ошибок.

Функция initialize_globals() инициализирует глобальные переменные, обнуляя массив PID, счётчик, ёмкость, структуру настроек терминала, строку пути к исполняемому файлу и флаг завершения.

Функция enable_raw_mode() переключает терминал в "сырой" режим, отключая эхо, каноническую обработку и сигналы от клавиш, таких как Ctrl+C, сохраняя исходные настройки терминала для последующего восстановления. Функция disable_raw_mode() восстанавливает исходные настройки терминала, обеспечивая корректное завершение программы. Функция cleanup_resources(), зарегистрированная через atexit(), освобождает ресурсы при завершении: восстанавливает терминал, завершает все дочерние процессы через kill_all_children() и освобождает память массива PID.

Функция handle_signal() обрабатывает сигналы SIGINT, SIGTERM, SIGQUIT, устанавливая флаг g_terminate_flag для завершения программы, а также сигнал SIGCHLD, вызывая waitpid() для очистки завершившихся дочерних процессов.

Эта функция является асинхронно-безопасной. Функция register_signal_handlers() настраивает обработчики для сигналов SIGINT, SIGTERM, SIGQUIT, SIGCHLD и игнорирует сигналы SIGUSR1 и SIGUSR2 в родительском процессе.

Функция add_child_pid() добавляет PID нового дочернего процесса в динамический массив, расширяя его при необходимости с помощью realloc(), и завершает процесс в случае ошибки выделения памяти.

Функция remove_child_pid_at_index() удаляет PID из массива по

указанному индексу, сдвигая последующие элементы.

Функция kill_all_children() отправляет сигнал SIGKILL всем отслеживаемым дочерним процессам и удаляет их из массива при успешной отправке сигнала или при ошибке ESRCH, указывающей на уже завершённый процесс.

Функция signal_all_children() отправляет сигналы SIGUSR1 или SIGUSR2 всем дочерним процессам, сообщая об успехах и ошибках.

Функция spawn_child() создаёт новый дочерний процесс с помощью fork() и execv(), добавляет его PID в массив и сообщает об успехе.

Функция kill_last_child() завершает последний созданный дочерний процесс сигналом SIGKILL, удаляя его из массива при успехе или ошибке ESRCH.

Функция list_children() выводит в стандартный поток вывода PID родительского процесса и всех отслеживаемых дочерних процессов. Функция safe_write() обеспечивает надёжную запись в файловый дескриптор, обрабатывая прерывания (EINTR) и частичные записи.

Дочерний процесс (child.c)

Функция main() является точкой входа дочернего процесса. Она инициализирует глобальные переменные через initialize_globals(), выводит информацию о запуске, настраивает обработчики сигналов с помощью register_signal_handlers() и устанавливает таймер через setup_timer(). В основном цикле функция выполняет неатомарные обновления структуры g_shared_pair, пока не будет достигнуто количество итераций, заданное NUM_REPETITIONS. После завершения цикла функция выводит статистику в стандартный поток вывода, если это разрешено, и возвращает EXIT_SUCCESS или EXIT_FAILURE в случае ошибок.

Функция initialize_globals() инициализирует структуру g_shared_pair, счётчики состояний, флаги g_alarm_flag, g_repetitions_done и g_output_enabled, устанавливая последний по умолчанию в значение, разрешающее вывод.

Функция handle_alarm() обрабатывает сигнал SIGALRM, считывая текущее состояние структуры g_shared_pair, увеличивая соответствующий счётчик (g_count00, g_count01, g_count10 или g_count11), обновляя счётчик итераций g_repetitions_done и устанавливая флаг g_alarm_flag. Эта функция является асинхронно-безопасной.

Функция handle_usr_signals() обрабатывает сигналы SIGUSR1, включая флаг g_output_enabled, и SIGUSR2, отключая его, обеспечивая асинхронную безопасность.

Функция register_signal_handlers() настраивает обработчики сигналов SIGALRM, SIGUSR1 и SIGUSR2 с использованием sigaction(). Функция

setup_timer() конфигурирует одноразовый таймер с помощью setitimer(), который отправляет сигнал SIGALRM через интервал, заданный ALARM_INTERVAL_US.

2.3 Описание порядка сборки и использования

Сборка проекта выполняется с помощью утилиты make, которая компилирует исходные файлы parent.c и child.c, размещая объектные файлы и исполняемые файлы в подкаталогах build/debug для отладочной сборки или build/release для релизной сборки. Makefile поддерживает два режима компиляции. Отладочный режим, используемый по умолчанию при вызове make или make debug-build, включает флаги -g3 и -ggdb для добавления отладочной информации.

Релизный режим, активируемый командой make MODE=release, применяет оптимизацию с флагом -O2 и преобразует предупреждения в ошибки с помощью -Werror.

Доступные команды сборки включают make для сборки отладочной версии, make release-build для релизной версии, make clean для удаления всех скомпилированных файлов путём удаления каталога build, make run для сборки и запуска отладочной версии с установкой переменной CHILD_PATH на build/debug, make run-release для сборки и запуска релизной версии с установкой CHILD_PATH на build/release, а также make help для вывода справки по использованию. Исполняемые файлы parent и child размещаются в соответствующей поддиректории build.

Для запуска программы необходимо установить переменную окружения CHILD_PATH, указывающую на директорию с исполняемым файлом child, например: env CHILD_PATH=$(pwd)/build/debug ./build/debug/parent. Программа ожидает ввода команд в реальном времени и завершает работу при вводе команды 'q' или получении сигнала завершения.

3 РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ

```
lab03 git:(main) make run
Debug build complete in build/debug
Running DEBUG version build/debug/parent...
CHILD_PATH will be set to: '/home/sovok/My Programs/Code
Studies/ОСиСП/Везенков М.Ю./lab03/build/debug'
env CHILD_PATH='/home/sovok/My Programs/Code Studies/ОСиСП/Везенков
М.Ю./lab03/build/debug' build/debug/parent
Parent process started (PID: 20804).
Commands: '+' spawn, '-' kill last, 'l' list, 'k' kill all,
         '1' enable child output (SIGUSR1), '2' disable child output
(SIGUSR2),
         'q' quit.
Using child executable: /home/sovok/My Programs/Code Studies/ОСиСП/Везенков
М.Ю./lab03/build/debug/child

PARENT [20804]: Spawned child process with PID 20805. Total children: 1
CHILD [20805]: Started. PPID=20804. Output initially ENABLED. Will run
10001 reps.

PARENT [20804]: Spawned child process with PID 20806. Total children: 2
CHILD [20806]: Started. PPID=20804. Output initially ENABLED. Will run
10001 reps.

PARENT [20804]: Listing processes:
 Parent: 20804
 Tracked Children (2):
   - PID 20805 (tracked)
   - PID 20806 (tracked)

PARENT [20804]: Sending SIGKILL to last child (PID 20806).
PARENT [20804]: Removed tracking for child 20806. Remaining children: 1

PARENT [20804]: Sending SIGKILL to last child (PID 20805).
PARENT [20804]: Removed tracking for child 20805. Remaining children: 0
```

Рисунок 3.1 – пример создания и убийства двух дочерних процессов

```
make run
Debug build complete in build/debug
Running DEBUG version build/debug/parent...
CHILD_PATH will be set to: '/home/sovok/My Programs/Code
Studies/ОСиСП/Везенков М.Ю./lab03/bui
ld/debug'
env CHILD_PATH='/home/sovok/My Programs/Code Studies/ОСиСП/Везенков
М.Ю./lab03/build/debug' bui
ld/debug/parent
Parent process started (PID: 20968).
Commands: '+' spawn, '-' kill last, 'l' list, 'k' kill all,
         '1' enable child output (SIGUSR1), '2' disable child output
(SIGUSR2),
         'q' quit.
Using child executable: /home/sovok/My Programs/Code Studies/ОСиСП/Везенков
М.Ю./lab03/build/de
bug/child

PARENT [20968]: No children to send SIGUSR2 (disable output) to.

PARENT [20968]: Spawned child process with PID 20970. Total children: 1
CHILD [20970]: Started. PPID=20968. Output initially ENABLED. Will run
10001 reps.

PARENT [20968]: Sending SIGUSR2 (disable output) to all 1 children.
PARENT [20968]: Attempted to send SIGUSR2 (disable output) to 1 children.
Success: 1, Already E
xited: 0.
CHILD [20970]: Final statistics output suppressed by signal.
CHILD [20970]: Exiting normally.
```

Рисунок 3.2 – пример создания дочернего процессов с выключенным выводом

4 ВЫВОД

В процессе выполнения лабораторной работы были изучены механизмы взаимодействия процессов в UNIX-подобных операционных системах, включая управление процессами, работу с сигналами и настройку терминала в режиме реального времени. В ходе работы была разработана программа, состоящая из родительского и дочернего процессов.

Родительский процесс управляет созданием, завершением и мониторингом дочерних процессов на основе команд, вводимых с клавиатуры, а дочерний процесс выполняет неатомарные обновления структуры данных, фиксируя их состояния при прерываниях от таймера, и выводит статистику в соответствии с заданными сигналами.