Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра электронных вычислительных машин

Лабораторная работа №3
по курсу
«Операционные системы и системное программирование»
на тему
«Взаимодействие и синхронизация процессов»

Выполнил: студент группы 350501
Русак Г.Д.
Проверил: старший преподаватель каф. ЭВМ
Поденок Л.П.

Минск 2025

1 ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

1.1 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.

1.2 Требования к сборке

Программы компилируются с ключами

```
-W -Wall -Wextra -std=c11 -pedantic
```

Допускается использование ключей

```
-Wno-unused-parameter -Wno-unused-variable.
```

Для компиляции, сборки и очистки используется `make`.

2 ВЫПОЛНЕНИЕ РАБОТЫ

2.1 Описание алгоритма выполнения работы

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

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

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

2.2 Алгоритм работы родительского процесса

Выделяется память для хранения информации о дочерних процессах.

Настраиваются обработчики сигналов (`SIGUSR1`, `SIGUSR2`, `SIGALRM`).

Основной цикл:

Программа ожидает ввода пользователя.

В зависимости от введённой команды выполняются соответствующие действия:

1) «+» - создаётся новый дочерний процесс.
2) «-» - удаляется последний созданный дочерний процесс.
3) «l» - выводится информация о всех дочерних процессах.
4) «k» - удаляются все дочерние процессы.
5) «s» - останавливает вывод данных для одного или всех дочерних процессов.
6) «g» - возобновляет вывод данных для одного или всех дочерних процессов.
7) «q» - завершает программу, удаляя все дочерние процессы и освобождая ресурсы.

Обработка сигналов:

`SIGUSR1` используется для остановки вывода данных дочернего процесса.

`SIGUSR2` используется для возобновления вывода данных дочернего процесса.

`SIGALRM` используется для автоматического возобновления работы всех дочерних процессов через определённое время.

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

2.3 Алгоритм работы дочернего процесса

Настраиваются обработчики сигналов (`SIGUSR1`, `SIGUSR2`, `SIGALRM`).

Устанавливается случайный таймер для обновления статистики.

В каждом цикле обновляется статистика (количество пар 00, 01, 10, 11).

Если прошло более 5 секунд и вывод разрешён, дочерний процесс отправляет статистику родительскому процессу через сигнал `SIGUSR1`.

Ожидает разрешения от родительского процесса для вывода данных.

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

Обработка сигналов:

1) `SIGUSR1` - запрещает вывод данных.

2) `SIGUSR2` - разрешает вывод данных.

3) `SIGALRM` - обновляет статистику и устанавливает новый таймер.

2.4 Взаимодействие между процессами

Создание дочернего процесса происходит следующим образом:

Родительский процесс создаёт дочерний процесс с помощью `fork()` и `execl()`. Информация о новом дочернем процессе 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.

Передача статистики:

Дочерний процесс отправляет статистику родительскому процессу через сигнал `SIGUSR1`. Родительский процесс обрабатывает сигнал и выводит статистику, если вывод разрешён.

3 ФУНКЦИОНАЛЬНАЯ СТРУКТУРА ПРОЕКТА

Файл `child.c` является точкой входа для дочернего процесса. Он реализует взаимодействие с родительским процессом через сигналы.

Основные функции `child.c`:

Функция `main()` является главной функцией дочернего процесса. В ней инициализируется дочерний процесс и происолдит основная работа с ним.

Файл `parent.c` является точкой входа для родительского процесса. Он реализует управление дочерними процессами через сигналы.

Основные функции `parent.c`:

Функция `main()` является главной функцией родительского процесса. В ней инициализируется родительский процесс и происходит основная работа с ним.

Файл `globals.h` определяет общие определения и структуры данных для родительского и дочернего процессов. Он содержит константы типы данных и объявление глобальных переменных.

Типы данных, которые используются:

Структура `pair_t` хранит пару значений. Поля структуры:

1) `int first` — первое значение;

2) `int second` — второе значение.

Структура `process_info_t` хранит информацию о процессе. Поля структуры:

1) `pid_t pid` — идентификатор процесса;

2) `bool is_stopped` — флаг остановки вывода;

3) `char name[CHILD_NAME_LENGTH]` — имя процесса.

Перечисление `child_state_t` хранит состояние дочернего процесса. Поля структуры:

1) `WAITING` — ожидание разрешения;

2) `PRINT_ALLOWED` — вывод разрешен;

3) `PRINT_FORBIDDEN` — вывод запрещен.

Файл `parent_functions.c` содержит реализацию функций родительского процесса для управления дочерними процессами через сигналы.

Основные функции в `parent_functions.c`:

Функция `alarm_handler()` является обработчиком сигнала таймера Принимаемые параметры:

1) `int sig` — номер сигнала.

Функция `parent_signal_handler()` является обработчиком сигналов от дочерних процессов. Принимаемые параметры:

1) `int sig` — номер сигнала;

2) `siginfo_t* info` — информация о сигнале;

3) `void* context` — контекст (не используется).

Функция `init_parent()` инициализирует родительский процесс.

Функция `cleanup_parent()` завершает работу родительского процесса.

Функция `create_child()` создает новый дочерний процесс.

Функция `remove_last_child()` удаляет последний дочерний процесс.

Функция `remove_all_children()` удаляет все дочерние процессы.

Функция `list_processes()` просматривает дочерние процессы.

Функция `disable_all_output()` останавливает работу всех дочерних процессов.

Функция `enable_all_output()` включает работу всех дочерних процессов.

Функция `disable_child_output()` останавливает работу определенного дочернего процесса. Принимаемые параметры:

1) `int child_num` — номер процесса.

Функция `enable_child_output()` включает работу определенного дочернего процесса. Принимаемые параметры:

1) `int child_num` — номер процесса.

Функция `request_child_output()` запрашивает вывести статистику для определенного дочернего процесса. Принимаемые параметры:

1) `int child_num` — номер процесса.

Функция `handle_user_input()` обрабатывает ввод пользователя. Принимаемые параметры:

1) `const char* input` — введенная пользователем строка.

Функция `parent_main_loop()` является основным циклом программы для родительского процесса.

Файл `child_functions.c` содержит реализацию функций дочернего процесса. Он обрабатывает сигналы, собирает статистику и взаимодейтсвует с родительским процессом.

Основные функции `child_functions.c`:

Функция `child_signal_handler()` является обработчиком сигналов для дочернего процесса. Принимаемые параметры:

1) `int sig` — номер сигнала.

Функция `update_statistics()` обновляет статистику на основе текущих значений.

Функция `update_stats_cycle()` обновляет значение пар статистики по циклу.

Функция `print_safe()` выводит строку посимвольно. Принимаемые параметры:

1) `const char* str` — строка для вывода.

Функция `print_statistics()` формирует и выводит статистику.

Функция `request_output_permission()` запрашивает разрешение на вывод у родителя.

Функция `init_child()` инициализирует дочерний процесс.

Функция `child_main_loop()` является основным циклом программы для дочернего процесса.

4 ПОРЯДОК СБОРКИ И ЗАПУСКА ПРОЕКТА

Порядок сборки и запуска состоит в следующем:

1) Клонировать репозиторий, используя команду `git clone https://github.com/Everolfe/lab3-OSASP`, или разархивировать каталог с проектом;

2) Перейти в каталог с проектом `cd lab3-OSASP`, или `cd "Имя разархивированного каталога";`

3) Собрать проект используя `make`;

4) После сборки проекта можно использовать, прописав `./parent`

5 РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ

```
grusak@fedora:~/tar_working_dir/Русак Г.Д./lab03$ ./parent
Parent process started. PID: 6491
Available commands:
  + : Create child
  - : Remove last child
  l : List processes
  k : Kill all children
  s : Disable all output
  g : Enable all output
  s<num> : Disable output for child <num>
  g<num> : Enable output for child <num>
  p<num> : Request output from child <num>
  q : Quit
> +
Created C_1 (PID: 6542)
> l
------------------------------------------------------
Parent PID: 6491
C_processes (2):
1. C_1 (PID: 6542, running)
2. C_2 (PID: 6548, running)
------------------------------------------------------
> s
Disabled output for all children
> l
------------------------------------------------------
Parent PID: 6491
C_processes (2):
1. C_1 (PID: 6542, stopped)
2. C_2 (PID: 6548, stopped)
------------------------------------------------------
> g
Enabled output for all children
> l
------------------------------------------------------
Parent PID: 6491
C_processes (2):
1. C_1 (PID: 6542, running)
2. C_2 (PID: 6548, running)
------------------------------------------------------
[child_6542 pid: 6542 ppid: 6491] stats: 00=26 01=25 10=25
11=25
C_6542 finished output
[child_6548 pid: 6548 ppid: 6491] stats:00=26 01=25 10=25
11=25
C_6548 finished output
```