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
Е.С. Шахлан

Проверил: старший преподаватель каф. ЭВМ
Л.П. Поденок

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. 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:

- «+» – создаётся новый дочерний процесс;

- «-» – удаляется последний созданный дочерний процесс;

- «l» – выводится информация о всех дочерних процессах;

- «k» – удаляются все дочерние процессы;

- «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:

SIGUSR1 – запрещает вывод данных;

SIGUSR2 – разрешает вывод данных;

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

Структура `proc_stats` используется для хранения текущего состояния дочернего процесса.

Поля:

- `pid_t pid` – Pid процесса;
- `char name[40]` – имя процесса;
- `bool is_acceptable` – значение состояния.

Глобальные переменные:

- `proc_num` – текущее количество дочерних процессов;
- `*child_proc` – массив всех дочерних процессов.

Функция `sig_handler()` – обработчик сигнала `SIGALRM`, устанавливает флаг `alarm_received.`

Передаваемые параметры:

- `int sig` – номер сигнала;
- `siginfo_t *inf` — указатель на структуру с дополнительной информацией о сигнале;
- `void *ptr` — указатель на контекст исполнения.

Функция `option_s()` – обработчик сигнала `SIGUSR1` у родительского процесса.

Функция `option_g()` – обработчик сигнала `SIGUSR2` у родительского процесса.

Функция `handler_settings()` – настройка обработки сигналов.

Функция `create_process()` – создание одного дочернего процесса.

Функция `delete_last_proc()` – завершение последнего созданного дочернего процесса.

Функция `delete_all_process()` – завершение всех дочерних процессов.

Функция `show_all_process()` – вывод всех текущих дочерних процессов.

Функция `main()` – точка входа в программу. Игнорирует сигналы `SIGUSR1` и `SIGUSR2` в родительском процессе, выводит список доступных команд в бесконечном цикле:

1) Читает команды;

2) Обрабатывает команды.

Взаимодействие процессов:

Родительский процесс управляет дочерними через сигналы:

- `SIGUSR1` – `silent` режим (вывод отключен);

- `SIGUSR2` – `grant` режим (вывод включен);

- `SIGTERM` – завершение процесса.

Дочерние процессы периодически меняют свое состояние (`DataPair`) и выводят статистику по завершении

Примеры команд:

- «`+`» – создать дочерний процесс;

- «`-`» – удалить последний дочерний процесс;

- «`l`» – вывести список процессов;

- «`k`» – удалить все дочерние процессы;

- «`s`» – `silent` режим для всех;

- «`g`» – `grant` режим для всех;

- «`q`» – завершить программу.

4 ПОРЯДОК СБОРКИ И ЗАПУСКА

1) Перейти в каталог проекта:

```
$ cd 'tar_working_dir/Шахлан Е.С./lab03'
```

2) Собрать проект с помощью `make`:

```
$ make
```

3) Запустить программу:

```
$ ./parent
```

5 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

```
+
PID 6787(C_00) - created
Processes number = 1
+
PID 6793(C_01) - created
Processes number = 2
PPID - 6768
PID - 6787
00 - 0
01 - 1
10 - 1
11 - 1
Child process 6787 printed info successfully
+
PID 6797(C_02) - created
Processes number = 3
PPID - 6768
PID - 6787
00 - 1
01 - 1
10 - 1
11 - 1
Child process 6787 printed info successfully
lPPID - 6768
PID - 6793
00 - 0
01 - 1
10 - 0
11 - 1
Child process 6793 printed info successfully
PPID - 6768
PID - 6797
00 - 1
01 - 1
```

```
10 - 0
11 - 1
Child process 6797 printed info successfully
l
parent - 6768
child :
C_00 -    6787 acceptable
C_01 -    6793 acceptable
C_02 -    6797 acceptable
PPID - 6768
PID - 6787
00 - 1
01 - 1
10 - 1
11 - 1
Child process 6787 printed info successfully
PPID - 6768
PID - 6793
00 - 1
01 - 1
10 - 1
11 - 1
Child process 6793 printed info successfully
PPID - 6768
PID - 6797
00 - 1
01 - 1
10 - 1
11 - 1
Child process 6797 printed info successfully
g
PPID - 6768
PID - 6787
00 - 1
01 - 1
10 - 1
11 - 1
```

```
Child process 6787 printed info successfully
PPID - 6768
PID - 6793
00 - 1
01 - 1
10 - 1
11 - 1
Child process 6793 printed info successfully
PPID - 6768
PID - 6797
00 - 1
01 - 1
10 - 1
11 - 1
Child process 6797 printed info successfully
PPID - 6768
PID - 6787
00 - 1
01 - 1
10 - 1
11 - 1
Child process 6787 printed info successfully
PPID - 6768
PID - 6793
00 - 1
01 - 1
10 - 1
11 - 1
Child process 6793 printed info successfully
PPID - 6768
PID - 6797
00 - 1
01 - 1
10 - 1
11 - 1
Child process 6797 printed info successfully
PPID - 6768
```

```
PID - 6787
00 - 1
01 - 1
10 - 1
11 - 1
Child process 6787 printed info successfully
PPID - 6768
PID - 6793
00 - 1
01 - 1
10 - 1
11 - 1
Child process 6793 printed info successfully
qPPID - 6768
PID - 6797
00 - 1
01 - 1
10 - 1
11 - 1
Child process 6797 printed info successfully
q
Process 6797 killed
Processes number = 2
Process 6793 killed
Processes number = 1
Process 6787 killed
Processes number = 0
```