Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра ЭВМ

Дисциплина: Операционные системы и системное программирование

ОТЧЁТ
к лабораторной работе №3
на тему
Взаимодействие и синхронизация процессов.

Выполнил студент гр.230501 Лазовский И.А.

Проверил старший преподаватель кафедры ЭВМ
Поденок Л.П.

Минск 2024

1 УСЛОВИЕ ЛАБОРАТОРНОЙ РАБОТЫ.

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.

При вводе символа «s» P запрещает всем C_k выводить статистику (см. ниже).

При вводе символа «g» P разрешает всем C_k выводить статистику.

При вводе символов «s<num>» P запрещает C_<num> выводить статистику.

При вводе символов «g<num>» P разрешает C_<num> выводить статистику.

При вводе символов «p<num>» P запрещает всем C_k вывод и запрашивает C_<num> вывести свою статистику. По истечению заданного времени (5 с, например), если не введен символ «g», разрешает всем C_k снова выводить статистику.

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 срабатывании 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, если ему разрешено, 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.

Программа предназначена для запуска родительского процесса `parent`, процесс дает возможность работать с дочерними процессами через различные операции , которые обрабатываются при нажатии.

Основные библиотечные функции которые использовались в лабораторной — `execve()`, `kill()`, `fork()`, `getpid()`, `getppid()` , `signal()`.

2.1. Программа parent

Данную программу мы запускаем, для того чтобы запустить дочерний процесс. Созданы глобальные переменные такие как:

- `array_of_child_program_pid` — массив который хранит `pid` всех дочерних процессов .

- `count_of_child_process` — где хранится количество запущенных дочерних процессов.

Работа функции `main` которые использовались в `parent`:

В начале программы объявляются необходимые переменные и массив array_of_child_program_pid, который используется для хранения идентификаторов дочерних процессов.

Затем программа входит в бесконечный цикл while, в котором ожидает ввода команды от пользователя с помощью функции fgets.

Далее идет проверка введенной команды. В коде представлены следующие команды:

Команда q: завершает все дочерние процессы, убивая их сигналом SIGKILL, и завершает программу.

Команда +: создает новый дочерний процесс с помощью функции fork и вызывает execve для запуска программы ./child.

Команда -: убивает последний созданный дочерний процесс с помощью сигнала SIGKILL.

Команда l: выводит информацию о родительском процессе и созданных дочерних процессах.

Команда k: завершает все дочерние процессы, убивая их сигналом SIGKILL.

Команда s: отправляет сигнал SIGUSR1 указанному дочернему процессу или всем дочерним процессам.

Команда g: отправляет сигнал SIGUSR2 указанному дочернему процессу или всем дочерним процессам.

Команда p: отправляет сигнал SIGUSR1 указанному дочернему процессу и ожидает ввода с клавиатуры в течение 5 секунд. Если вводится символ "g", отправляется сигнал SIGUSR2 указанному дочернему процессу.

Для выполнения команды p, используется функция select, которая позволяет ожидать ввода на стандартный ввод (stdin) в течение определенного времени. Если происходит ввод в течение 5 секунд, программа проверяет введенный символ. Если это символ "g", отправляется сигнал SIGUSR2 указанному дочернему процессу.

После выполнения каждой команды программа возвращается в начало цикла while и ожидает новую команду от пользователя.

2.2. Программа child

Константы:

- `INTERVAL` - задает интервал для счетчика;

Структура данных:

- `struct pair` - структура для хранения пары чисел x и y;

Глобальные переменные:

- `pair`: Экземпляр структуры pair для хранения текущей пары чисел;
- `stat01, stat00, statll, statl0`: Переменные для хранения статистики;
- `counter` – счетчик для отслеживания числа итераций;
- `output_allowed` - флаг для разрешения/запрещения вывода статистики;

Обработчики сигналов:

- `alarm_handler` – Обработчик сигнала SIGALRM, который вызывается при срабатывании таймера. Обновляет статистику в зависимости от значения текущей пары чисел.
- `sig1_handler, sig2_handler` – Обработчики сигналов SIGUSR1 и SIGUSR2 соответственно. Они управляют флагом output_allowed, позволяя или запрещая вывод статистики.

Основная функция `main`():

- установка обработчиков сигналов.
- установка таймера сигнала `SIGALRM`.
- бесконечный цикл, в котором генерируются пары чисел и обновляется статистика.
- при достижении `counter` значения `INTERVAL`, программа выводит статистику, если вывод разрешен, и обнуляет счетчик и статистику.

3. ФУНКЦИОНАЛЬНАЯ СТРУКТУРА ПРОЕКТА.

Проект собирается с помощью makefile. Пример запуска:

```
ivan@fedora:~/Рабочий стол/lab3$ build/debug/parent
```

Для запуска проекта нам требуется в терминале заупустить программу parent. Где программа сразу переходит в цикл обработки символов

В проекте имеется каталог для сборки debug и release. Каталог git для системы контроля версий моего проекта. Директория src с исходным кодом. И makefile для компиляции и сборки проекта.

4. ПОРЯДОК СБОРКИ И ИСПОЛЬЗОВАНИЯ.

Для компиляции и сборки проекта используется makefile.

Порядок сборки:

1) Задание переменных:

- `DEBUG и RELEASE` - пути к каталогам для отладочной и релизной сборки соответственно.

- `OUT_DIR` - текущий каталог для выходных файлов (по умолчанию используется отладочная сборка).

- `FLAGS_DEBUG и CFLAGS_RELEASE` - флаги компиляции для отладочной и релизной сборки соответственно.

`object_child` и `object_parent` - объектные файлы для компиляции.

`Parent и child` - имя выходного исполняемого файла.

2. Определение компилятора и флагов компиляции:

`CC` - компилятор (gcc).

`CFLAGS` - флаги компиляции, выбираются в зависимости от переменной `MODE` (отладочная или релизная сборка).

3. Определение зависимостей:

`vpath` - указание директорий для поиска файлов с исходным кодом и заголовочных файлов.

`ifeq ($(MODE), release)` - установка флагов и каталогов в случае релизной сборки.

4. Определение целей:

`all` - основная цель, компиляция всех объектных файлов и создание исполняемого файла.

`$(child) и $(parent)` - правило для создания исполняемого файла.

`$(OUT_DIR)/%.o: %.c` - правило для компиляции каждого исходного файла в объектный.

Порядок использования.

1. Компиляция:

Для отладочной сборки: `make` или `make MODE=debug`

Для релизной сборки: `make MODE=release`

2. Очистка:

`make clean` - удаляет все объектные файлы и исполняемый файл.

3. Запуск:

После успешной компиляции запустите исполняемый файл, например: ./build/debug/parent.

5. МЕТОД ТЕСТИРОВАНИЯ И РЕЗУЛЬТАТ ТЕСТИРОВАНИЯ.

```
ilua@fedora:~/Рабочий стол/lab3$ build/debug/parent
+
---CREATED CHILD PROCESS WITH NUMBER 0---
PPID:13651 PID:13652 00:999 01:879 10:889 11:1011
s
SIGUSR1 received
+
---CREATED CHILD PROCESS WITH NUMBER 1---
PPID:13651 PID:13653 00:1131 01:995 10:967 11:1096
PPID:13651 PID:13653 00:1146 01:977 10:969 11:1111
g
SIGUSR2 received
SIGUSR2 received
PPID:13651 PID:13652 00:1126 01:941 10:937 11:1109
PPID:13651 PID:13653 00:1126 01:931 10:992 11:1123
PPID:13651 PID:13652 00:1145 01:952 10:898 11:1112
PPID:13651 PID:13653 00:1117 01:921 10:959 11:1182
s
SIGUSR1 received
SIGUSR1 received
l
Родительский процесс PID = 13651
Дочерний процесс 0 PID = 13652
Дочерний процесс 1 PID = 13653
g0
SIGUSR2 received
PPID:13651 PID:13652 00:1162 01:908 10:931 11:1124
PPID:13651 PID:13652 00:1106 01:938 10:927 11:1136
s
SIGUSR1 received
SIGUSR1 received
q
```

```
---ALL PROCESS DELETED---
```