Студент Крамаренко М.К., группа 234, ИДЗ №2, вариант №30

### Задание:

Военная операция. Темной–темной ночью прапорщики Иванов, Петров и Нечепорук занимаются хищением военного имущества со склада родной военной части. Будучи умными людьми и отличниками боевой и строевой подготовки, прапорщики ввели разделение труда. Иванов выносит имущество со склада и передает его в руки Петрову, который грузит его в грузовик. Нечепорук стоит на шухере и заодно подсчитывает рыночную стоимость добычи поле погрузки в грузовик очередной партии похищенного.

Требуется составить многопроцессное приложение, моделирующее деятельность прапорщиков–потоков.

Необходимо учесть случайное время выполнения каждым прапорщиком своей боевой задачи и организовать в программе корректную их синхронизацию.

**Обращаю внимание**, что в каждой программе есть файл launch.sh, который отвечает за компиляцию программ, а также выводит инструкцию по запуску программы. Цель скрипта ­– быстрый запуск программ.

# **Реализация на 4-5 балла. Программа находится в папке 04-05.**

**Разработать консольное приложение, в котором единый родительский процесс запускает требуемое число дочерних процессов.**

Требования:

1. **Представить сценарий решаемой задачи поясняющий, каким образом исходные сущности и их поведение отображаются в процессы и их взаимодействие.**

Общий сценарий решения:

1. Создать три семафора для трёх сущностей (прапорщиков). У семафора первой сущности будет 1 свободный ресурс, чтобы его сразу можно было использовать, у остальных семафоров изначально не будет свободных ресурсов.
2. Создать для каждой сущности отдельный процесс при помощи **fork()**.
3. Каждый процесс будет ждать свободный ресурс своего семафора. Из п. 1, первый процесс получит свой ресурс сразу, остальные будет ждать.
4. Когда первый процесс получил свой ресурс, он имитирует действие первой сущности. После завершения действия, освобождает ресурс второго процесса.
5. Когда второй процесс получил свой ресурс, он имитирует действие второй сущности. После завершения действия, освобождает ресурс третьего процесса.
6. Когда третий процесс получил свой ресурс, он имитирует действие третьей сущности. После завершения действия, освобождает ресурс первого процесса. Далее переходим к п. 4.
7. **Способ обмена данными: Множество процессов взаимодействуют с использованием именованных POSIX семафоров. Обмен данными ведется через разделяемую память в стандарте POSIX.**

Для создания именованных семафоров используется функция **sem\_open()**.

Для создания разделяемой памяти используется функция **shm\_open()**.

1. **Реализовать завершение программы в соответствии с условием задачи, а также предусмотреть корректное завершение по прерыванию с клавиатуры по соответствующему сигналу.**

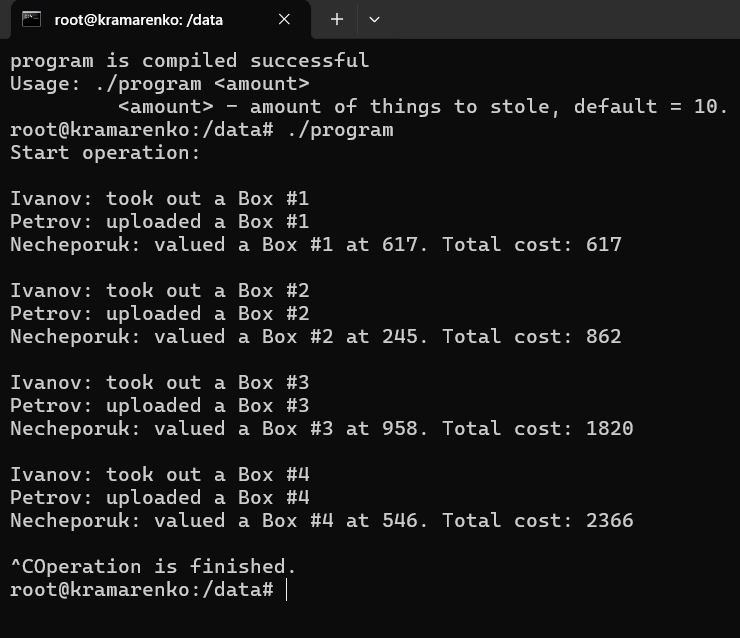
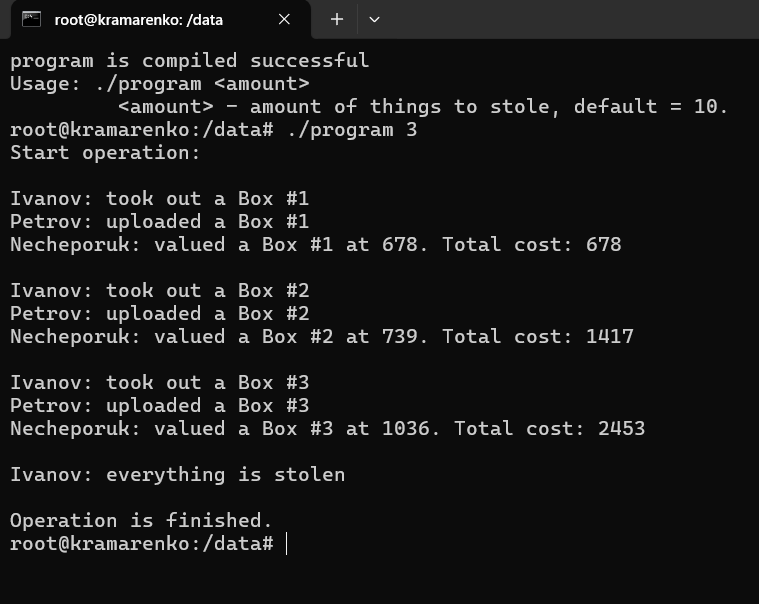
Программа завершается, когда все вещи были украдены. Количество вещей для кражи устанавливает пользователь, по умолчанию стоит 10 вещей.

Также программа перехватывает сигнал **SIGINT** (**Ctrl+C**) и также завершает программу.

1. **В программе предусмотреть удаление семафоров и разделяемой памяти по ее завершению любым из способов.**

Для корректного освобождения ресурсов используется функция **dipose()**.

1. **Пример работы программы.**



Запуск программы с параметром.

Запуск программы без параметров и прерывание программы при помощи **Ctrl+C**.

# **Реализация на 6-7 баллов. Программа находится в папке 06-07.**

**В дополнение к программе на предыдущую оценку необходимо разработать программу, в которой для взаимодействия процессов используется один из двух вариантов, описанных выше, но не реализованных на предшествующую оценку.**

**Т.е. Способ обмена данными: Множество процессов взаимодействуют с использованием неименованных POSIX семафоров. Обмен данными ведется через разделяемую память в стандарте POSIX.**

Для создания неименованных семафоров используется функция **sem\_init()**, для освобождения ресурсов семафора – **sem\_destroy()**. Остальная логика программы не изменилась.

# **Реализация на 8 баллов. Программа находится в папке 08.**

**В дополнение к программам, выполненным на предыдущие оценки, необходимо разработать приложение, состоящее из отдельных программ–процессов, запускаемых независимо друг от друга.**

Требования:

1. **Для завершения приложения использовать те же подходы, что и в предыдущих решениях.**

Программы корректно завершаются.

1. **Способ обмена данными: Множество независимых процессов взаимодействуют с использованием семафоров в стандарте UNIX SYSTEM V. Обмен данными ведется через разделяемую память в стандарте UNIX SYSTEM V.**

Для создания семафоров используется функция **semget()**, для освобождения ресурсов семафора – **semctl(semid, 0, IPC\_RMID, 0)**.

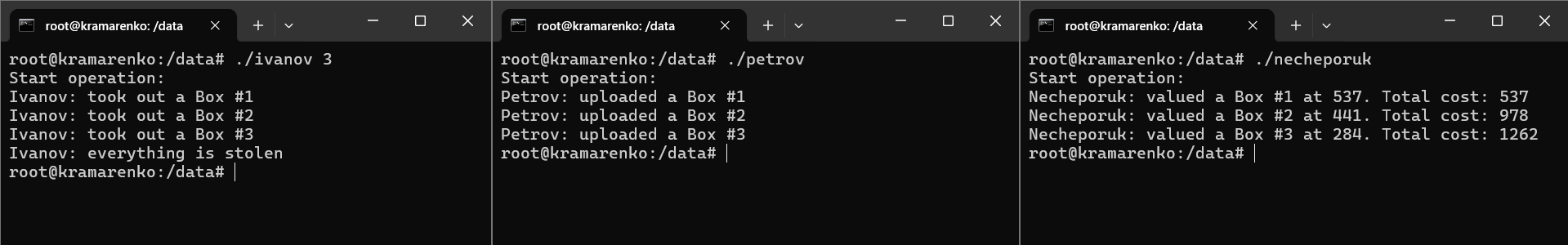
Для создания разделяемой памяти используется функция **shmget()**, для освобождения ресурсов разделяемой памяти – **shmctl()**.

1. **Информация о добавленной реализации и пример работы.**

Теперь существует 3 программы (ivanov.c, petrov.c, necheporuk.c).

Главная программа ivanov.c, поэтому она запускается первой. Затем запускаются остальные программы, порядок не важен.

Пример запуска:



# **Реализация на 9 баллов. Программа находится в папке 09.**

**В дополнение к программам на предыдущие оценки необходимо разработать приложение, состоящее из отдельных программ–процессов, запускаемых независимо друг от друга. Использовать семафоры и каналы (или очереди сообщений) для организации взаимодействия между процессами.**

Требования:

1. **Для завершения приложения использовать те же подходы, что и в предыдущих решениях.**

Программы корректно завершаются.

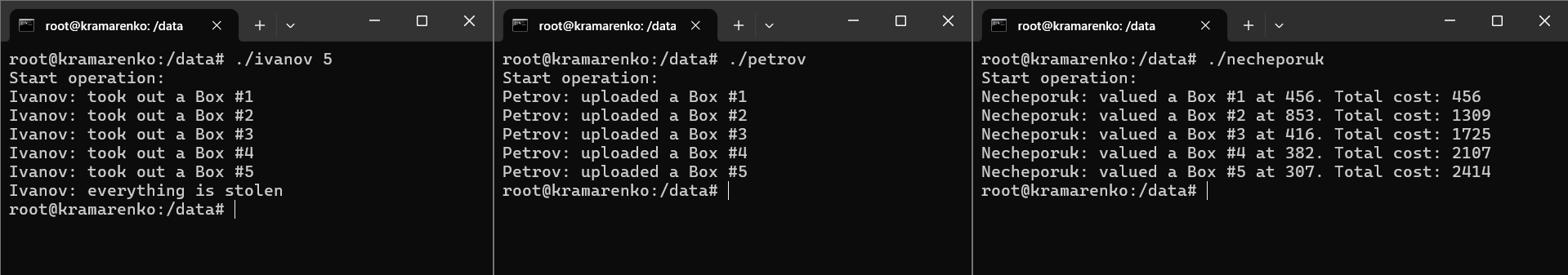
1. **Способ обмена данными: Множество независимых процессов взаимодействуют с использованием именованных POSIX семафоров. Обмен данными ведется через каналы или очереди сообщений в стандарте POSIX.**

* Программы используют 3 именованных POSIX семафора, для инициализации которого используется функция **sem\_open()**.
* Все 3 программы используют один именованный канал, для инициализации которого используется функция **mkfifo()**.
* Доступ к каналу синхронизируется при помощи семафоров.

1. **Информация о добавленной реализации и пример работы.**

Для упрощения работы с каналами, программа ivanov.c передаёт только наименование украденной вещи, а третья программа necheporuk.c оценивает вещь (скорее всего так было задумано изначально).

Пример работы программ:



# **Реализация на 10 баллов. Программа находится в папке 10.**

**В дополнение к программам на предыдущие оценки необходимо разработать программу, в которой для взаимодействия процессов используется оставшийся вариант, из описанных на 9 баллов.**

**Т.е. Способ обмена данными: Множество независимых процессов взаимодействуют с использованием семафоров в стандарте UNIX SYSTEM V. Обмен данными ведется через каналы или очереди сообщений в стандарте UNIX SYSTEM V.**

Логика программы осталась прежней.

Вместо POSIX канала используется очередь сообщений в стандарте System Unix V (функция **semget()**).

Вместо POSIX семафоров используются семафоры в стандарте System Unix V (функция **msgget()**).