Министерство образования и науки Российской Федерации

Федеральное агентство по образованию

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

**Основы системного программирования**

**Лабораторная работа №5**

Студенты: Мелихова К.А.

Гаспаров В.М.

Филоненко В.С.

Группа: №R3330

САНКТ-ПЕТЕРБУРГ

2019

**Задание**

**Часть 1**

Сервер при запуске должен сохранить:

* идентификатор процесса;
* идентификатор пользователя процесса;
* идентификатор группы процесса;

и раз в секунду обновлять:

* количество секунд, прошедших с момента запуска процесса;
* среднюю загрузку системы за последние 1, 5 и 15 минут.

Сервер должен быть одним процессом с одним потоком.

Клиент должен один раз выводить текущую информацию от сервера.

Для получения необходимой информации будут полезны man страницы:  
getpid(2), getuid(2), getgid(2), time(2), getloadavg(3C).

Задания этой части должны быть выполнены только на языке С. Подзадания:

1. System V shared memory

Для хранения требуемой информации и обмена информацией между клиентом и сервером используется разделяемый сегмент памяти. man shmget(2), shmat(2), shmdt(2).

1. System V message queue

Клиент обменивается с сервером используя System V message queue. man msgget(2), msgsnd(2), msgrcv(2).

1. Отображение файла в память

Для хранения требуемой информации и обмена информацией между клиентом и сервером используется файл, отображённый в память с использованием mmap(2).

**Часть 2**

Процесс должен иметь заполненный массив, последовательно заполненный буквами английского алфавита в нижнем регистре. Процесс должен запустить два потока, один из которых умеет инвертировать регистр букв, второй - разворачивает массив, т.е. делает первую букву последней, вторую - предпоследней и т.д.

Задания этой части должны быть выполнены только на языке С. Подзадания:

1. Основной поток раз в секунду поочерёдно пробуждает один из потоков и дожидается конца его работы, после чего выводит полученный массив на стандартный вывод. Для реализации синхронизации необходимо использовать sem\_init(3RT), sem\_post(3RT), sem\_wait(3RT).
2. Основной поток раз в секунду поочерёдно пробуждает один из потоков и дожидается конца его работы, после чего выводит полученный массив на стандартный вывод. Для реализации синхронизации необходимо использовать semget(2), semop(2), semctl(2).
3. Основной поток раз в указанное количество микросекунд выводит массив на стандартный вывод. Подпотоки раз в указанное количество микросекунд выполняют своё действие, блокируя на время своей работы доступ к общему ресурсу. Для блокировки использовать pthread\_mutex\_init(3C), pthread\_mutex\_lock(3C), pthread\_mutex\_unlock(3C).
4. Основной поток раз в указанное количество микросекунд выводит массив на стандартный вывод. Подпотоки раз в указанное количество микросекунд выполняют своё действие. Еще один подпоток должен раз в указанное количество микросекунд выводить количество заглавных символов в массиве на стандартный вывод. Потоки, не изменяющие массив, могут выполняться параллельно, но должны блокировать изменение на время работы. Для блокировки использовать pthread\_rwlock\_init(3C), pthread\_rwlock\_rdlock(3C), pthread\_rwlock\_wrlock(3C), pthread\_rwlock\_unlock(3C).

**Часть 3**

Задания этой части должны быть выполнены и на языке С, и на языке perl. Подзадания:

1. Unix domain socket

Клиент обменивается с сервером используя unix domain socket (AF\_UNIX). man socket(3SOCKET), bind(3SOCKET), listen(3SOCKET), accept(3SOCKET), connect(3SOCKET), select(3C).

Сервер должен соответствовать требованиям первой части.

1. Обработка сигналов

При получении сигналов HUP, INT, TERM, USR1, USR2 сервер в стандартный поток вывода выводит соответствующий пункт накопленной информации. man signal(3C), sigaction(2), kill(1).

Сервер должен соответствовать требованиям первой части.

1. Неименованные каналы

Процесс запускает дочерний подпроцесс, которому подменяется стандартный ввод. В подпроцессе запускается системная утилита wc. Основной процесс читает указанный через аргументы файл и передаёт в подпроцесс каждый чётный символ. man fork(2), execl(2), pipe(2), dup2(3C). При использовании system(3C) или popen(3C) задание принято не будет.

**Исходный код**

[**https://github.com/frosty322/Osp5and6/tree/master/src**](https://github.com/frosty322/Osp5and6/tree/master/src)

[**https://github.com/frosty322/Osp5and6/tree/master/perl**](https://github.com/frosty322/Osp5and6/tree/master/perl)