**Содержание**

[**1. Цели работы** 2](#_Toc88766062)

[**2. Задачи** 2](#_Toc88766063)

[**3. Ход выполнения работы** 3](#_Toc88766064)

[**4. Вывод** 7](#_Toc88766065)

# **Лабораторная работа №9 – Работа с разделяемой памятью**

# **1. Цели работы**

Использование для обмена данными разделяемой памяти (shared memory) - самого быстрого средства межпроцессорного взаимодействия в Linux.

**2. Задачи**

1. Войти в систему и скопировать с разделяемого ресурса в свой HOME-каталог набор исходных файлов для десятого занятия.

2. Скомпилировать и выполнить программу gener\_shm.cpp, которая демонстрирует создание сегментов разделяемой памяти. Запустить программу несколько раз и выполнить команду ipcs -m.

3. Удалить созданные на предыдущем шаге сегменты разделяемой памяти с помощью команды ipcrm с соответствующей опцией и значением id сегмента или ключа.

4. Скомпилировать shmdemo.cpp, осуществляющую операции записи в разделяемую память без разделения доступа к этому общему ресурсу. Символы, записываемые в общую память, передаются в качестве параметра командной строки при запуске процесса shmdemo. Запуск этого процесса без параметров приводит к выводу на консоль текущего содержимого сегмента общей памяти.

5. Запустить несколько раз процессы типа shmdemo с различными значениями параметров и проиллюстрируйте возможности чтения и записи в сегмент общей памяти независимо исполняемыми процессами. Затем удалить сегмент памяти командой ipcrm.

6. Скомпилировать и выполните программу attach\_shm.cpp, иллюстрирующую передачу символьной информации между двумя процессами (родственными) через сегмент общей памяти с модификацией этой информации.

7. Составить программу, создающую три разделяемых сегмента памяти размером 1023 байта каждый. Укажите в вызове shmat() параметр shmaddr = 0 при привязке сегментов.

**3. Ход выполнения работы**

1. Войду в систему и скопировать с разделяемого ресурса в свой HOME-каталог набор исходных файлов для десятого занятия.

2. Скомпилирую и выполню программу gener\_shm.cpp, которая демонстрирует создание сегментов разделяемой памяти. Запустить программу несколько раз и выполнить команду ipcs -m. Результаты приведены на рисунке 9.1 и 9.2. Видно, что создается два сегмента с id 22 и 23, а во втором случае с 22 и 24. Важно уточнить, что первый участок создается по идентификатору 22, а после системный вызов возвращает его идентификатор. Второй участок создается каждый раз, так как его значение key равно IPC\_PRIVATE.

|  |
| --- |
| Рис. 9.1 Результат выполнения программы gener\_shm.cpp |

3. Удалю созданные на предыдущем шаге сегменты разделяемой памяти с помощью команды ipcrm с соответствующей опцией и значением id сегмента или ключа. Результаты представлены на рисунке 9.2.

|  |
| --- |
| Рис. 9.2 – Удаление разделяемой памяти |

4. Скомпилирую shmdemo.cpp, осуществляющую операции записи в разделяемую память без разделения доступа к этому общему ресурсу. Символы, записываемые в общую память, передаются в качестве параметра командной строки при запуске процесса shmdemo. Запуск этого процесса без параметров приводит к выводу на консоль текущего содержимого сегмента общей памяти. Важно отметить, что программа создает участок памяти равный 1024 байтам. Выделенный участок памяти представлен на рисунке 9.3 под id равным 42.

|  |
| --- |
| Рис. 9.3 – Создание участка памяти |

5. Запущу несколько раз процессы типа shmdemo с различными значениями параметров и проиллюстрирую возможности чтения и записи в сегмент общей памяти независимо исполняемыми процессами. Затем удалю сегмент памяти командой ipcrm. Как видно из полученных изображений сначала программа получает доступ к разделяемому участку памяти по ключу. После проходит вызов shamat(), который подключается к памяти с полученным идентификатором. После введение строки, программа или записывает или выводит строку. На рисунке 9.4 приведено записывание строки в разделяемую память. В завершении удаляю сегмент выделенной памяти (рисунок 9.5)

|  |
| --- |
| Рис. 9.4 – Запись строки в память |
| 9.5 – Удаление выделенного сегмента |

6. Скомпилирую и выполню программу attach\_shm.cpp, иллюстрирующую передачу символьной информации между двумя процессами (родственными) через сегмент общей памяти с модификацией этой информации. Как видно из полученных данных, приведенных на рисунке 9.6, в сегмент памяти записывается английский алфавит, после процесс потомок изменяет ее, а именно изменяет регистр на нижний. Процесс родитель выводит измененные данные на экран и завершает работу. В программе происходит вызов shmdt(), который отделяет сегменты разделяемой памяти от процессов. После завершения процесса потомка, родитель помечает сегмент как удаленный с помощью вызова shmctl() и пользователь уже не может к нему обратиться.

|  |
| --- |
| Рис. 9.6 – Выполнение программы attach |

7. Составлю программу, создающую три разделяемых сегмента памяти размером 1023 байта каждый. Укажите в вызове shmat() параметр shmaddr = 0 при привязке сегментов. Важно отметить, что при обращении к 1024 сегменту памяти происходит ошибка. А также сегменты расположились по соседству, что и следовало проверить в задании. Результаты выполнения программы приведены на рисунке 9.7. Выделенные сегменты обозначены id равными 41, 42, 43.

|  |
| --- |
| Рис. 9.7 – Выполнение программы lab |

**4. Вывод**

В ходе выполнения программы было на практике изучено выделение и работа с разделяемой памятью. Были созданы сегменты, также производилась запись и чтение на сегменты разделяемой памяти. Был продемонстрирован процесс передачи информации между двумя процессами через сегмент памяти.