РАБОТА С РАЗДЕЛЯЕМОЙ ПАМЯТЬЮ

Цель работы.

Использование для обмена данными разделяемой памяти (shared memory) – самого быстрого средства меж процессного взаимодействия в Linux.

1. Скомпилируйте и выполните программу gener\_shm.cpp демонстрирующую создание сегментов разделяемой памяти. Запустите программу несколько раз и после каждого ее завершения выполните команду ipcs -m . Поясните зависимость процедуры создания сегментов разделяемой памяти от используемых в вызове shmget() флагов.

|  |
| --- |
|  |
| Рис. 9-1 Многократное исполнение программы gener\_shm |

Первый участок памяти создаётся по ключу 15. После, системный вызов shmget() лишь возвращает его идентификатор, найденный по ключу.

Второй участок создаётся каждый раз так-как значение key равно **IPC\_PRIVATE**.

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

|  |
| --- |
|  |
| Рис. 9-2 Удаление сегментов разделяемой памяти |

3. Скомпилируйте 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.

|  |
| --- |
|  |
| Рис. 9-3 Создание сегментов программой shmdemo |

Несложно заметить, что программа создаёт новый участок разделяемой памяти размером 1 килобайт.

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

|  |
| --- |
| Изображение выглядит как текст  Автоматически созданное описание |
| Рис. 9-4 Демонстрация работы shmdemo |

При запуске программа получает идентификатор к сегменту разделяемой памяти по своему ключу. После происходит вызов shamat(), который подключается сегмент общей памяти с нашим идентификатором к адресному пространству вызывающего процесса. После в зависимости от ввода программа или записывает или выводит строку. В конце происходит отключение сегмента общей памяти от процесса.

|  |
| --- |
|  |
| Рис. 9-5 Удаление сегмента |

5. Скомпилируйте и выполните программу attach\_shm.cpp , иллюстрирующую передачу символьной информации между двумя процессами (родственными) через сегмент общей памяти с модификацией этой информации. Проанализируйте значения выводимой информации о границах сегментов в системной памяти. За счет чего после завершения данной программы сегмент общей памяти уже не присутствует в системе?

|  |
| --- |
|  |
| Рис. 9-6 Выполнение программы из 5 пункта |

В ходе выполнения программы с сегмент общей памяти записывается английский алфавит, после процесс-потомок изменяет ее (переводя буквы в нижний регистр), а процесс-родитель выводит эти данные и удаляет сегмент памяти.

То есть оба потомка отстыковываются сегменты разделяемой памяти от процессов через вызов shmdt(). А процесс-родитель (который подождал 3 секунды для завершения процесса-потомка) при помощи shmctl() помечает сегмент как удалённый. И поэтому после выполнения программы мы не можем пронаблюдать сегмент разделённой памяти.

6. Составьте программу, создающую три разделяемых сегмента памяти размером 1023 байта каждый. Укажите в вызове shmat() параметр shmaddr = 0 при привязке сегментов. Разместит ли система сегменты в последовательных участках? Позволит ли система ссылку или изменение 1024-го байта любого из этих участков?

|  |
| --- |
|  |
| Рис. 9-7 Создание трёх сегментов |

Сегменты действительно разместились “по соседству” в памяти. При обращении к 1024 биту любого из них получаем ошибку.