Кафедра прикладной математики и кибернетики

Выполнили:

Студенты 3 курса группы ИП-111  
Корнилов А.А.,  
Попов М.И.,

Толкач А.А.

Проверил:

Профессор кафедры ПМиК  
Малков Е.А.

ЛАБОРАТОРНАЯ РАБОТА №12

По дисциплине: «Операционные системы»

Сибирский государственный университет  
телекоммуникаций и информатики

Новосибирск, 2023

Министерство цифрового развития, связи  
и массовых коммуникаций Российской Федерации

**Задание:** Протестируйте код лекции 12.

**Цель:** знакомство с выделением памяти, разделяемой разными процессами, и ее синхронизацией.

**Выполнение работы:**

Суть программы lab13b.c в открытие файла, отображение в его в память, перемешивании символов в файле, его сохранении и закрытии.

|  |
| --- |
| #include <stdlib.h> #include <stdio.h> #include <fcntl.h> #include <sys/stat.h> #include <time.h> #include <unistd.h> #include <sys/mman.h>  int main(int argc**,** char \*const argv[]) {  int fd**;** struct stat stat\_file**;** char dummy**;** char \*map\_address**;** fd = open("test.txt"**,** O\_RDWR | O\_CREAT**,** S\_IRUSR | S\_IWUSR | S\_IRGRP)**;** if (fd == -1)  fprintf(stderr**,** "open\n")**;** if (fstat(fd**,** &stat\_file)) // Получаем информацию о файле (размер, права и т.д.)  fprintf(stderr**,** "fstat\n")**;** // Открываем файл "test.txt" с флагами O\_RDWR (открытие для чтения и записи) и O\_CREAT (создание, если не существует)  // и устанавливаем права доступа S\_IRUSR (чтение для владельца), S\_IWUSR (запись для владельца), S\_IRGRP (чтение для группы)  map\_address = (char \*) mmap(0**,** stat\_file.st\_size**,** PROT\_READ | PROT\_WRITE**,** MAP\_PRIVATE**,** fd**,** 0)**;** if (map\_address == MAP\_FAILED)  fprintf(stderr**,** "mmap\n")**;** //меняем "This is a test." на "Is this a test?"  dummy = map\_address[1]**;** map\_address[0] = map\_address[5] - 0x20**;** map\_address[1] = map\_address[3]**;** map\_address[2] = map\_address[4]**;** map\_address[3] = map\_address[10]**;** map\_address[4] = dummy**;** map\_address[14] = 63**;** write(fd**,** map\_address**,** stat\_file.st\_size)**;** // Записываем измененные данные обратно в файл  munmap(map\_address**,** stat\_file.st\_size)**;** // Освобождаем отображенную область файла из памяти   close(fd)**;** return 0**;** } |

Листинг 1 – программа lab13b.c c расставленными комментариями

Команда компиляции и результат работы программы:

|  |
| --- |
| miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$ cat test.txt  This is a test.  miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$ gcc lab13b.c -o lab13b  miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$ ./lab13b  miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$ cat test.txt  Is this a test?  miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$ |

В программе lab13c-1.c (листинг 2) в открываем файл, отображаем в его в память с общим доступом (флаг MAP\_SHARED), записываем «Take it easy!», ожидаем ввода пользователя и после нажатия сохраняем и закрываем файл.

Программа lab13c-2.c (листинг 3) также открывает файл «test\_shared.txt», но если запустить lab13c-1 и во время её выполнения и ожидания ввода пользователя открыть lab13c-2 программа выведет «Take it easy!» которая уже записана файл, а так как файл открыть для общего доступа то запись и вывод доступны разным процессам одновременно.

|  |
| --- |
| #include <stdio.h> #include <fcntl.h> #include <sys/stat.h> #include <unistd.h> #include <string.h> #include <sys/mman.h>  int main(int argc**,** char \*const argv[]) {   int fd**;** struct stat stat\_file**;** char \*map\_address**;** // Открываем файл "test\_shared.txt"  fd = open("test\_shared.txt"**,** O\_RDWR | O\_CREAT**,** S\_IRUSR | S\_IWUSR | S\_IRGRP)**;** if (fd == -1)  fprintf(stderr**,** "open\n")**;** // Отображаем файл в память с правами PROT\_READ (чтение), PROT\_WRITE (запись)  // и флагом MAP\_SHARED (создаем общее отображение файла)  map\_address = (char \*) mmap(0**,** 256**,** PROT\_READ | PROT\_WRITE**,** MAP\_SHARED**,** fd**,** 0)**;** if (map\_address == MAP\_FAILED)  fprintf(stderr**,** "mmap\n")**;** close(fd)**;** // Копируем строку "Take it easy!" в отображенную область файла  memcpy(map\_address**,** "Take it easy!\0"**,** sizeof("Take it easy!\0"))**;** getc(stdin)**;** // Освобождаем отображенную область файла из памяти  munmap(map\_address**,** 256)**;** return 0**;** } |

Листинг 2 – программа lab13c-1.c c расставленными комментариями

|  |
| --- |
| #include <fcntl.h> #include <stdio.h> #include <sys/stat.h> #include <unistd.h> #include <string.h> #include <sys/mman.h>  int main(int argc**,** char \*const argv[]) {   int fd**;** struct stat stat\_file**;** char \*map\_address**;** fd = open("test\_shared.txt"**,** O\_RDWR)**;** if (fd == -1)  fprintf(stderr**,** "open\n")**;** map\_address = (char \*) mmap(0**,** 256**,** PROT\_READ | PROT\_WRITE**,** MAP\_SHARED**,** fd**,** 0)**;** if (map\_address == MAP\_FAILED)  fprintf(stderr**,** "mmap\n")**;** close(fd)**;** //выводим на экран содержимое test\_shared.txt  write(fileno(stdout)**,** map\_address**,** 256)**;** getc(stdin)**;** munmap(map\_address**,** 256)**;** return 0**;** } |

Листинг 2 – программа lab13c-2.c c расставленными комментариями

Команда компиляции и результат работы программы:

|  |
| --- |
| miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$ gcc lab13c-1.c -o lab13c-1  miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$ gcc lab13c-2.c -o lab13c-2  miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$ ./lab13c-1  */\*------ другой терминал -------\*/*  miron@DESKTOP-UMC1Q46:/$ cat proc/1789/maps  55fc1082c000-55fc1082d000 r--p 00000000 00:53 2533274790806769 /mnt/u/Documents/В ВУЗ/OS/12/lab13c-1  …  55fc1082d000-55fc1082e000 r-xp 00001000 00:53 2533274790806769  55fc11ba7000-55fc11bc8000 rw-p 00000000 00:00 0 [heap]  7f882d0cd000-7f882d0d0000 rw-p 00000000 00:00 0  …  **7f882d337000-7f882d338000 rw-s 00000000 00:53 1125899907253494 /mnt/u/Documents/В ВУЗ/OS/12/test\_shared.txt**  …  miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$ ./lab13c-2  Take it easy! |

В программе lab13d-1.c (листинг 3) мы созадем общее пространство памяти, отображаем в его в память с общим доступом (флаг MAP\_SHARED), записываем «Take it easy! Be happy!», ожидаем ввода пользователя и после нажатия закрываем файл. Программа lab13d-2.c открывает эту память и выводит на экран её содержимое.

|  |
| --- |
| #include <stdio.h> #include <fcntl.h> #include <sys/stat.h> #include <unistd.h> #include <string.h> #include <sys/mman.h>  int main(int argc**,** char \*const argv) {   int fd**;** char \*map\_address**;** // Создаем или открываем общую область памяти с именем "/common\_region"  // с флагами O\_RDWR (открытие для чтения и записи), O\_CREAT (создание, если не существует)  // и устанавливаем права доступа S\_IRUSR (чтение для владельца), S\_IWUSR (запись для владельца), S\_IRGRP (чтение для группы)  fd = shm\_open("/common\_region"**,** O\_RDWR | O\_CREAT**,** S\_IRUSR | S\_IWUSR | S\_IRGRP)**;** if (fd == -1) fprintf(stderr**,** "open \n")**;** ftruncate(fd**,** 256)**;** // Устанавливаем размер общей области памяти в 256 байт   // Отображаем общую область памяти в адресное пространство процесса  // с правами PROT\_READ (чтение), PROT\_WRITE (запись) и флагом MAP\_SHARED (создаем общее отображение)  map\_address = (char) mmap(0**,** 256**,** PROT\_READ | PROT\_WRITE**,** MAP\_SHARED**,** fd**,** 0)**;** if (map\_address == MAP\_FAILED) fprintf(stderr**,** "mmap\n")**;** close(fd)**;** //копируем "Take it easy! Be happy!" в общую память  memcpy(map\_address**,** "Take it easy! Be happy!\0"**,** sizeof("Take it easy! Be happy!\0"))**;** getc(stdin)**;** munmap(map\_address**,** 256)**;** //очищаем память  shm\_unlink("/common\_region")**;** //разыменовываем область памяти   return 0**;** } |

Листинг 3 – программа lab13d-1.c c расставленными комментариями

|  |
| --- |
| #include <stdio.h> #include <fcntl.h> #include <sys/stat.h> #include <unistd.h> #include <string.h> #include <sys/mman.h>  int main(int argc**,** char \*const argv[]) {  int fd**;** char \*map\_address**;** fd = shm\_open("/common\_region"**,** O\_RDWR**,** S\_IRUSR | S\_IWUSR | S\_IRGRP)**;** if (fd == -1) fprintf(stderr**,** "shm\_open\n")**;** map\_address = (char \*) mmap(0**,** 256**,** PROT\_READ | PROT\_WRITE**,** MAP\_SHARED**,** fd**,** 0)**;** if (map\_address == MAP\_FAILED)  fprintf(stderr**,** "mmap\n")**;** close(fd)**;** write(fileno(stdout)**,** map\_address**,** 256)**;** getc(stdin)**;** munmap(map\_address**,** 256)**;** return 0**;** } |

Листинг 4 – программа lab13d-2.c

Команда компиляции и результат работы программы:

|  |
| --- |
| miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$ gcc lab13d-1.c -lrt -o lab13d-1  miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$ gcc lab13d-2.c -lrt -o lab13d-2  miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$ ./lab13d-1  */\*------ другой терминал -------\*/*  miron@DESKTOP-UMC1Q46:/$ cat proc/1875/maps  56160c3db000-56160c3dc000 r--p 00000000 00:53 7036874418177264 /mnt/u/Documents/В ВУЗ/OS/12/lab13d-1  …  56160d92a000-56160d94b000 rw-p 00000000 00:00 0 [heap]  7fbf53f3a000-7fbf53f3d000 rw-p 00000000 00:00 0  …  **7fbf541a4000-7fbf541a5000 rw-s 00000000 00:3d 2 /dev/shm/common\_region**  miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$ ./lab13d-2  Take it easy! Be happy! |

Программы lab13e-1.c (листинг 5) и lab13e-2.c (листинг 6) подобны ранее описанным, но в них используется семафоры для последовательного доступа к памяти. Для удобства цикл был заменен на while(1)

|  |
| --- |
| #include <semaphore.h> #include <sys/stat.h> #include <fcntl.h> #include <stdio.h> #include <string.h> #include <unistd.h> #include <sys/mman.h>  int main(void) {  int n = 0**;** int fd**;** char \*sh**;** sem\_t \*sem**;** // Создаем или открываем именованную область памяти с именем "/common\_region"  // с флагами O\_RDWR (открытие для чтения и записи), O\_CREAT (создание, если не существует)  // и устанавливаем права доступа S\_IRUSR (чтение для владельца), S\_IWUSR (запись для владельца), S\_IRGRP (чтение для группы)  fd = shm\_open("/common\_region"**,** O\_RDWR | O\_CREAT**,** S\_IRUSR | S\_IWUSR | S\_IRGRP)**;** if (fd == -1) fprintf(stderr**,** "shm\_open\n")**;** ftruncate(fd**,** 6)**;** // Устанавливаем размер общей области памяти в 6 байт   // Отображаем общую область памяти в адресное пространство процесса  // с правами PROT\_READ (чтение), PROT\_WRITE (запись) и флагом MAP\_SHARED (создаем общее отображение)  sh = (char \*) mmap(0**,** 6**,** PROT\_READ | PROT\_WRITE**,** MAP\_SHARED**,** fd**,** 0)**;** if (sh == MAP\_FAILED) fprintf(stderr**,** "mmap\n")**;** memset(sh**,** 0**,** 6)**;** sem = sem\_open("/common\_sem"**,** O\_CREAT**,** S\_IRUSR | S\_IWUSR | S\_IRGRP**,** 1)**;** // 🚦 не работает  if (sem == SEM\_FAILED) fprintf(stderr**,** "sem\_open")**;** while (1) {  sem\_wait(sem)**;** // Ожидаем, когда 🚦 станет доступным (уменьшаем его значение на 1)  //write(fileno(stdout),sh, 6);  printf("String: %s\n"**,** sh)**;** // Выводим содержимое общей области с использованием функции printf  sem\_post(sem)**;** // Освобождаем семафор (увеличиваем его значение на 1)  usleep(100)**;** }   // Удаляем именованную область памяти, память, именованный 🚦 и его самого  shm\_unlink("/common\_region")**;** munmap(sh**,** 6)**;** sem\_unlink("/common\_sem")**;** sem\_close(sem)**;** return 0**;** } |

Листинг 5 – программа lab13e-1.c

|  |
| --- |
| #include <semaphore.h> #include <sys/stat.h> #include <fcntl.h> #include <stdio.h> #include <string.h> #include <unistd.h> #include <sys/mman.h>  int main() {  int n = 0**;** int counter = 0**;** int fd**;** char \*sh**;** sem\_t \*sem**;** fd = shm\_open("/common\_region"**,** O\_RDWR | O\_CREAT**,** S\_IRUSR | S\_IWUSR | S\_IRGRP)**;** sh = (char \*) mmap(0**,** 6**,** PROT\_READ | PROT\_WRITE**,** MAP\_SHARED**,** fd**,** 0)**;** memset(sh**,** 0**,** 6)**;** //находим 🚦  sem = sem\_open("/common\_sem"**,** 0)**;** while (n++ < 200) {  sem\_wait(sem)**;** //смотрим на 🚦  if (counter % 2) { //пишем Hello или Bye\_u  sh[0] = 'H'**;** sh[1] = 'e'**;** sh[2] = 'l'**;** sh[3] = 'l'**;** sh[4] = 'o'**;** sh[5] = '\0'**;** } else {  sh[0] = 'B'**;** sh[1] = 'y'**;** sh[2] = 'e'**;** sh[3] = '\_'**;** sh[4] = 'u'**;** sh[5] = '\0'**;** }  sem\_post(sem)**;** counter++**;** usleep(100)**;** }  munmap(sh**,** 6)**;** return 0**;** } |

Листинг 6 – программа lab13e-2.c

Команда компиляции и результат работы программы:

|  |
| --- |
| miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$ gcc lab13e-2.c -o lab13e-2  miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$ gcc lab13e-1.c -o lab13e-1  miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$./lab13e-1  String:  String:  String:  String:  String:  String:  String:  …  */\*------ другой терминал -------\*/*  miron@DESKTOP-UMC1Q46:/$ cat proc/1979/maps  55776b9a6000-55776b9a7000 r--p 00000000 00:53 3659174697649416 /mnt/u/Documents/В ВУЗ/OS/12/lab13e-1  …  55776d3aa000-55776d3cb000 rw-p 00000000 00:00 0 [heap]  7f1f2b0c4000-7f1f2b0c7000 rw-p 00000000 00:00 0  …  7f1f2b2e2000-7f1f2b2ef000 rw-p 00000000 00:00 0  **7f1f2b2f4000-7f1f2b2f5000 rw-s 00000000 00:3d 8 /dev/shm/sem.dqZEtq (deleted)**  …  **7f1f2b32e000-7f1f2b32f000 rw-s 00000000 00:3d 7 /dev/shm/common\_region**  miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$ ./lab13e-2  */\*------ переключаемся обратно -------\*/*  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  String: Hello  … (через >~200)  String: Bye\_u  String: Bye\_u  String: Bye\_u  String: Bye\_u  String: Bye\_u  String: Bye\_u  String: Bye\_u  String: Bye\_u  String: Bye\_u  String: Bye\_u  String: Bye\_u  String: Bye\_u  String: Bye\_u  String: Bye\_u  … |

Программы lab13em-1.c (листинг 7) и lab13em-2.c (листинг 8) синхронизируют доступ к памяти при помощи mutex-ов.

|  |
| --- |
| #include <pthread.h> #include <sys/stat.h> #include <fcntl.h> #include <stdio.h> #include <string.h> #include <unistd.h> #include <sys/mman.h>  int main(void) {  int n = 0**;** int fd**;** char \*sh**;** pthread\_mutex\_t \*Mutex**;** // Указатель на мьютекс (синхронизация доступа к общей памяти)  pthread\_mutexattr\_t mutex\_attr**;** // Атрибуты мьютекса   // Создаем или открываем именованную область памяти с именем "/common\_region1"  // с флагами O\_RDWR (открытие для чтения и записи), O\_CREAT (создание, если не существует)  // и устанавливаем права доступа S\_IRUSR (чтение для владельца), S\_IWUSR (запись для владельца), S\_IRGRP (чтение для группы)  fd = shm\_open("/common\_region1"**,** O\_RDWR | O\_CREAT**,** S\_IRUSR | S\_IWUSR | S\_IRGRP)**;** if (fd == -1) fprintf(stderr**,** "shm\_open\n")**;** ftruncate(fd**,** 6)**;** // Устанавливаем размер общей области памяти в 6 байт   // Отображаем общую область памяти в адресное пространство процесса  // с правами PROT\_READ (чтение), PROT\_WRITE (запись) и флагом MAP\_SHARED (создаем общее отображение)  sh = (char \*) mmap(0**,** 6**,** PROT\_READ | PROT\_WRITE**,** MAP\_SHARED**,** fd**,** 0)**;** if (sh == MAP\_FAILED) fprintf(stderr**,** "mmap\n")**;** close(fd)**;** memset(sh**,** 0**,** 6)**;** // Заполняем общую область памяти нулями   fd = shm\_open("/common\_mutex"**,** O\_RDWR | O\_CREAT**,** S\_IRUSR | S\_IWUSR | S\_IRGRP)**;** if (fd == -1) fprintf(stderr**,** "shm\_open for mutex\n")**;** ftruncate(fd**,** sizeof(pthread\_mutex\_t))**;** // Устанавливаем размер области памяти для мьютекса в размер pthread\_mutex\_t   pthread\_mutexattr\_init(&mutex\_attr)**;** // Инициализируем атрибуты мьютекса  pthread\_mutexattr\_setpshared(&mutex\_attr**,** PTHREAD\_PROCESS\_SHARED)**;** // Устанавливаем атрибут PTHREAD\_PROCESS\_SHARED, чтобы мьютекс был видим в разных процессах  // Отображаем область памяти для мьютекса в адресное пространство процесса  Mutex = (pthread\_mutex\_t \*) mmap(0**,** sizeof(pthread\_mutex\_t)**,** PROT\_READ | PROT\_WRITE**,** MAP\_SHARED**,** fd**,** 0)**;** close(fd)**;** pthread\_mutex\_init(Mutex**,** &mutex\_attr)**;** // Инициализируем мьютекс с использованием атрибутов   while (1) {  pthread\_mutex\_lock(Mutex)**;** // Захватываем мьютекс  //write(fileno (stdout), sh, 6);  printf("String: %sin"**,** sh)**;** pthread\_mutex\_unlock(Mutex)**;** // Освобождаем мьютекс  }   munmap(sh**,** 6)**;** munmap(Mutex**,** sizeof(pthread\_mutex\_t))**;** shm\_unlink("/common\_mutex")**;** shm\_unlink("/common\_region1")**;** getc(stdin)**;** return 0**;** } |

Листинг 7 – программа lab13em-1.c

|  |
| --- |
| #include <pthread.h> #include <sys/stat.h> #include <fcntl.h> #include <stdio.h> #include <string.h> #include <unistd.h> #include <sys/mman.h>  int main() {  int n = 0**;** int counter = 0**;** int fd**;** char \*sh**;** pthread\_mutex\_t \*Mutex**;** fd = shm\_open("/common\_region1"**,** O\_RDWR | O\_CREAT**,** S\_IRUSR | S\_IWUSR | S\_IRGRP)**;** sh = (char \*) mmap(0**,** 6**,** PROT\_READ | PROT\_WRITE**,** MAP\_SHARED**,** fd**,** 0)**;** close(fd)**;** memset(sh**,** 0**,** 6)**;** fd = shm\_open("/common\_mutex"**,** O\_RDWR | O\_CREAT**,** S\_IRUSR | S\_IWUSR | S\_IRGRP)**;** Mutex = (pthread\_mutex\_t \*) mmap(0**,** sizeof(pthread\_mutex\_t)**,** PROT\_READ | PROT\_WRITE**,** MAP\_SHARED**,** fd**,** 0)**;** close(fd)**;** while (1) {  pthread\_mutex\_lock(Mutex)**;**// Захватываем мьютекс  if (counter % 2) { //пишем или Hello или Bye\_u  sh[0] = 'H'**;** sh[1] = 'e'**;** sh[2] = 'l'**;** sh[3] = 'l'**;** sh[4] = 'o'**;** sh[5] = '\0'**;** } else {  sh[0] = 'B'**;** sh[1] = 'y'**;** sh[2] = 'e'**;** sh[3] = '\_'**;** sh[4] = 'u'**;** sh[5] = '\0'**;** }  pthread\_mutex\_unlock(Mutex)**;** counter++**;** }  getc(stdin)**;** munmap(sh**,** 6)**;** shm\_unlink("/common\_region1")**;** munmap(Mutex**,** sizeof(pthread\_mutex\_t))**;** shm\_unlink("/common\_mutex")**;** return 0**;** } |

Листинг 8 – программа lab13em-2.c

Команда компиляции и результат работы программы:

|  |
| --- |
| miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$ gcc lab13em-1.c -o lab13em-1  miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$ gcc lab13em-2.c -o lab13em-2  miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$ ./lab13em-1  String : String : String : String : String : String : String : String : String : String : String : String : String : String : String : String : String : String : String : String : String : String : String : String : String : String : String : String : String : String : String : String : String : String :…  */\*------ другой терминал -------\*/*  miron@DESKTOP-UMC1Q46:/$ cat proc/1921/maps  555acd6a4000-555acd6a5000 r--p 00000000 00:53 6473924464755621 /mnt/u/Documents/В ВУЗ/OS/12/lab13em-1  …  555ace102000-555ace123000 rw-p 00000000 00:00 0 [heap]  7f3866b63000-7f3866b66000 rw-p 00000000 00:00 0  …  7f3866d81000-7f3866d8e000 rw-p 00000000 00:00 0  **7f3866d93000-7f3866d94000 rw-s 00000000 00:3d 4 /dev/shm/common\_mutex**  …  **7f3866dcd000-7f3866dce000 rw-s 00000000 00:3d 3 /dev/shm/common\_region1**  **…**  miron@DESKTOP-UMC1Q46:/mnt/u/Documents/В ВУЗ/OS/12$ ./lab13em-2  */\*------ переключаемся обратно -------\*/*  String: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: HelloinString: Bye\_uinString: Bye\_uinString: HelloinString: HelloinString: Bye\_uinString: HelloinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: Bye\_uinString: … |