Министерство цифрового развития, связи
и массовых коммуникаций Российской Федерации

Сибирский государственный университет
телекоммуникаций и информатики

Кафедра прикладной математики и кибернетики

ЛАБОРАТОРНАЯ РАБОТА №12

По дисциплине: «Операционные системы»

Выполнили:

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

Проверил:

Профессор кафедры ПМиК
Малков Е.А.

Новосибирск, 2023

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.

```c
#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 с расставленными комментариями

Команда компиляции и результат работы программы:

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

Команда компиляции и результат работы программы:

```
miron@DESKTOP-UMC1Q46:/mnt/u/Documents/B ВУЗ/OS/12$ gcc lab13c-1.c -o
lab13c-1
miron@DESKTOP-UMC1Q46:/mnt/u/Documents/B ВУЗ/OS/12$ gcc lab13c-2.c -o
lab13c-2
miron@DESKTOP-UMC1Q46:/mnt/u/Documents/B 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/B 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/B ВУЗ/OS/12/test_shared.txt
…
miron@DESKTOP-UMC1Q46:/mnt/u/Documents/B ВУЗ/OS/12$ ./lab13c-2
Take it easy!
```

В программе lab13d-1.c (листинг 3) мы создем 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>
```

```c
#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 с расставленными комментариями

```c
#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/B ВУЗ/OS/12$ gcc lab13d-1.c -lrt -o
lab13d-1
miron@DESKTOP-UMC1Q46:/mnt/u/Documents/B ВУЗ/OS/12$ gcc lab13d-2.c -lrt -o
lab13d-2
miron@DESKTOP-UMC1Q46:/mnt/u/Documents/B 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/B 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/B 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>
```

```c
#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

```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/BУЗ/OS/12$ gcc lab13e-2.c -o
lab13e-2
```

```
miron@DESKTOP-UMC1Q46:/mnt/u/Documents/B ВУЗ/OS/12$ gcc lab13e-1.c -o
lab13e-1
miron@DESKTOP-UMC1Q46:/mnt/u/Documents/B 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/B 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/B 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
String: Bye_u
…
```

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

```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(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/B  ВУЗ/OS/12$  gcc  lab13em-1.c  -o
lab13em-1
miron@DESKTOP-UMC1Q46:/mnt/u/Documents/B  ВУЗ/OS/12$  gcc  lab13em-2.c  -o
lab13em-2
miron@DESKTOP-UMC1Q46:/mnt/u/Documents/B 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 :…

/*------ другой терминал -------*/

miron@DESKTOP-UMC1Q46:/$ cat proc/1921/maps
555acd6a4000-555acd6a5000    r--p    00000000    00:53    6473924464755621
/mnt/u/Documents/B 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/B 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: …
```