**Задание 1** Создайте файл hello.txt с текстом Hello, mmap!. Используя [заготовку](/programming-languages/cse-programming-languages-fall-2023/main/-/blob/master/seminar-4/hello_mmap.asm), отобразите его в память и выведите текст из него в стандартный поток вывода. Не забудьте вызвать munmap (его номер системного вызова 11) и закрыть файл (close, номер системного вызова 3) по завершению работы с файлом.

**hello\_mmap.asm:**

%define O\_RDONLY 0

%define PROT\_READ 0x1

%define MAP\_PRIVATE 0x2

%define SYS\_WRITE 1

%define SYS\_OPEN 2

%define SYS\_MMAP 9

%define FD\_STDOUT 1

%define PAGE\_SIZE 4096

section .data

    ; This is the file name. You are free to change it.

    fname: db 'hello.txt', 0

section .text

global \_start

; use exit system call to shut down correctly

exit:

    mov  rax, 60

    xor  rdi, rdi

    syscall

; These functions are used to print a null terminated string

; rdi holds a string pointer

print\_string:

    push rdi

    call string\_length

    pop  rsi

    mov  rdx, rax

    mov  rax, SYS\_WRITE

    mov  rdi, FD\_STDOUT

    syscall

    ret

string\_length:

    xor  rax, rax

.loop:

    cmp  byte [rdi+rax], 0

    je   .end

    inc  rax

    jmp .loop

.end:

    ret

; This function is used to print a substring with given length

; rdi holds a string pointer

; rsi holds a substring length

print\_substring:

    mov  rdx, rsi

    mov  rsi, rdi

    mov  rax, SYS\_WRITE

    mov  rdi, FD\_STDOUT

    syscall

    ret

\_start:

    ; Вызовите open и откройте fname в режиме read only.

    mov  rax, SYS\_OPEN

    mov  rdi, fname

    mov  rsi, O\_RDONLY    ; Open file read only

    mov  rdx, 0           ; We are not creating a file

                          ; so this argument has no meaning

    syscall

    ; rax holds the opened file descriptor now

    ; Вызовите mmap c правильными аргументами

    ; Дайте операционной системе самой выбрать, куда отобразить файл

    ; Размер области возьмите в размер страницы

    ; Область не должна быть общей для нескольких процессов

    ; и должна выделяться только для чтения.

    mov r12, rax ; save file descriptor

    mov rax, 9              ; mmap syscall

    mov rdi, 0              ; OS will choose mapping address

    mov rsi, PAGE\_SIZE

    mov rdx, PROT\_READ

    mov r10, MAP\_PRIVATE

    mov r8, r12             ; file descriptor

    mov r9, 0               ; offset

    syscall

    mov r12, rax

    mov rdi, rax

    call print\_string

    mov rax, 11             ; munmap syscall

    mov rdi, r12

    mov rsi, PAGE\_SIZE

    syscall

    mov rax, 2              ; close file syscall

    mov rdi, fname

    syscall

    call exit

**Результат:**

**Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**

**Задание 2** Прочитайте документацию по системному вызову fstat (номер 5). Вас будет интересовать поле st\_size типа off\_t структуры struct stat, которую функция fstat заполняет. Используйте его, чтобы корректно вычислить размер файла при выводе данных, и выведите их, используя функцию print\_substring (принимает на вход два аргумента: адрес начала строки и количество байт для вывода). Полученный размер файла используйте в вызовах mmap, munmap и print\_substring.

**hello\_mmap.asm**

%define O\_RDONLY 0

%define PROT\_READ 0x1

%define MAP\_PRIVATE 0x2

%define SYS\_WRITE 1

%define SYS\_OPEN 2

%define SYS\_MMAP 9

%define FD\_STDOUT 1

%define PAGE\_SIZE 4096

section .data

    ; This is the file name. You are free to change it.

    fname: db 'hello.txt', 0

section .text

global \_start

; use exit system call to shut down correctly

exit:

    mov  rax, 60

    xor  rdi, rdi

    syscall

; These functions are used to print a null terminated string

; rdi holds a string pointer

print\_string:

    push rdi

    call string\_length

    pop  rsi

    mov  rdx, rax

    mov  rax, SYS\_WRITE

    mov  rdi, FD\_STDOUT

    syscall

    ret

string\_length:

    xor  rax, rax

.loop:

    cmp  byte [rdi+rax], 0

    je   .end

    inc  rax

    jmp .loop

.end:

    ret

; This function is used to print a substring with given length

; rdi holds a string pointer

; rsi holds a substring length

print\_substring:

    mov  rdx, rsi

    mov  rsi, rdi

    mov  rax, SYS\_WRITE

    mov  rdi, FD\_STDOUT

    syscall

    ret

\_start:

    ; Вызовите open и откройте fname в режиме read only.

    mov  rax, SYS\_OPEN

    mov  rdi, fname

    mov  rsi, O\_RDONLY      ; Open file read only

    mov  rdx, 0             ; We are not creating a file

                            ; so this argument has no meaning

    syscall

    ; rax holds the opened file descriptor now

    mov r12, rax            ; save file descriptor

    sub rsp, 144            ; allocate memory for fstat struct

    mov rax, 5              ; fstat syscall

    mov rdi, r12

    mov rsi, rsp

    syscall

    mov r13, [rsp + 48]     ; get "size" fstat struct field

    mov rax, 9              ; mmap syscall

    mov rdi, 0              ; OS will choose mapping address

    mov rsi, PAGE\_SIZE

    mov rdx, PROT\_READ

    mov r10, MAP\_PRIVATE

    mov r8, r12             ; file descriptor

    mov r9, 0               ; offset

    syscall

    mov rdi, rax

    mov rsi, r13

    call print\_substring

    add rsp, 144

    mov rax, 11             ; munmap syscall

    mov rdi, r12

    mov rsi, PAGE\_SIZE

    syscall

    mov rax, 2              ; close file syscall

    mov rdi, fname

    syscall

    call exit

**Результат:**

**Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**

**Задание 3** В этих файлах не хватает нескольких строчек чтобы можно было взаимодействовать с кодом друг друга. Допишите файлы так, чтобы функции print\_string и hello вызывалась и проверьте результат. Подсказка: вспомните, что нужно, чтобы из одного файла с C-кодом вызвать код из другого файла.

**Структура:**

**Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**

**asm\_headers.h**

#ifndef ASM\_HEADERS

#define ASM\_HEADERS

void print\_string(char\* str);

void world();

#endif

**hello.c**

#include "asm\_headers.h"

int main() {

    print\_string("hello");

    world();

}

**string.asm**

global world

global print\_string

section .rodata

message: db ", world!", 10, 0

section .text

string\_length:

    mov  rax, 0

    .loop:

      xor  rax, rax

    .count:

      cmp  byte [rdi+rax], 0

      je   .end

      inc  rax

      jmp  .count

    .end:

      ret

world:

    mov rdi, message

print\_string:

    push rax

    push rdi

    call string\_length

    mov  rsi, [rsp]

    mov  rdx, rax

    mov  rax, 1

    mov  rdi, 1

    push rcx

    syscall

    pop  rcx

    pop  rdi

    pop  rax

    ret

**Результат:**

**Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**

**Задание 4** Объедините ассемблерный код для вывода содержимого файла с кодом на языке C. Пусть ваша программа будет просить пользователя ввести название файла, а затем выведет его содержимое в стандартный поток вывода используя код, написанный в начале сегодняшнего семинара (сделайте из него функцию print\_file, которая будет принимать имя файла первым аргументом). Не забудьте, что для корректной работы необходимо следовать соглашениям о вызовах и сохранить callee-saved регистры, которые вы используете, в начале своей функции print\_file. Для вывода сообщений (например “Please enter file name: “) используйте собственную реализацию print\_string из сегодняшнего семинара.

**asm\_headers.h**

#ifndef ASM\_HEADERS

#define ASM\_HEADERS

void print\_file(char\* filename);

void print\_string(char\* str);

#endif

**main.c**

#include <stdio.h>

#include "asm\_headers.h"

int main() {

    char input[255];

    print\_string("Please, enter filename: ");

    if (scanf("%s", input) > 0) {

        putchar('\n');

        print\_file(input);

        putchar('\n');

        return 0;

    }

    else {

        return 1;

    }

}

**lib.asm**

%define O\_RDONLY 0

%define PROT\_READ 0x1

%define MAP\_PRIVATE 0x2

%define SYS\_WRITE 1

%define SYS\_OPEN 2

%define SYS\_MMAP 9

%define FD\_STDOUT 1

%define PAGE\_SIZE 4096

section .text

global print\_file

global print\_string

; use exit system call to shut down correctly

exit:

    mov  rax, 60

    xor  rdi, rdi

    syscall

; These functions are used to print a null terminated string

; rdi holds a string pointer

print\_string:

    push rdi

    call string\_length

    pop  rsi

    mov  rdx, rax

    mov  rax, SYS\_WRITE

    mov  rdi, FD\_STDOUT

    syscall

    ret

string\_length:

    xor  rax, rax

.loop:

    cmp  byte [rdi+rax], 0

    je   .end

    inc  rax

    jmp .loop

.end:

    ret

; This function is used to print a substring with given length

; rdi holds a string pointer

; rsi holds a substring length

print\_substring:

    push r12

    push r13

    push r14

    mov  rdx, rsi

    mov  rsi, rdi

    mov  rax, SYS\_WRITE

    mov  rdi, FD\_STDOUT

    syscall

    pop r14

    pop r13

    pop r13

    ret

print\_file:

    mov r14,  rdi           ; save filename pointer

    mov  rax, SYS\_OPEN

    mov  rsi, O\_RDONLY      ; Open file read only

    mov  rdx, 0             ; We are not creating a file

                            ; so this argument has no meaning

    syscall

    ; rax holds the opened file descriptor now

    mov r12, rax            ; save file descriptor

    sub rsp, 144            ; allocate memory for fstat struct

    mov rax, 5              ; fstat syscall

    mov rdi, r12

    mov rsi, rsp

    syscall

    mov r13, [rsp + 48]     ; get "size" fstat struct field

    mov rax, 9              ; mmap syscall

    mov rdi, 0              ; OS will choose mapping address

    mov rsi, PAGE\_SIZE

    mov rdx, PROT\_READ

    mov r10, MAP\_PRIVATE

    mov r8, r12             ; file descriptor

    mov r9, 0               ; offset

    syscall

    mov rdi, rax

    mov rsi, r13

    call print\_substring

    mov rax, 11             ; munmap syscall

    mov rdi, r12

    mov rsi, PAGE\_SIZE

    syscall

    mov rax, 2              ; close file syscall

    mov rdi, r14

    syscall

.end:

    add rsp, 144

    xor rax, rax

    ret

**Результат:**

**Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**