Языки программирования (Asm)

Занятие №3. Работа с файлами.

МИРЭА - РТУ, БК №252

Москва 2020

Создание файла

Для работы с файлами нам понадобится еще несколько системных вызовов ядра Linux. Первый из них **sys_creat**, который используется для создания нового файла.

```
1  section .data:
2  filename db "test_file.txt", 0
3
4  section .text:
5    global _start
6  _start:
7    mov eax, 8 ; sys_creat
8    mov ebx, filename
9    mov ecx, 420
10   int 0x80
11   mov [fd], eax ; Сохраняем файловый дескриптор в переменную
```

В случае успешного создания файла, файловый дескриптор будет автоматически помещен в регистр EAX.

Права доступа и закрытие файла

```
1 mov ecx, 420
```

Данной строкой мы задали права доступа к новому файлу. В данном случае, это права -**rw-r–r–**. Т.е. право на чтение для всех и на запись только для владельца файла. Если перевести число 420 в двоичный вид, то получим 110100100. Единицы стоят на тех местах, где указано соответствующее право.

Для закрытия файла используется системный вызов **sys_close**. Через регистр EBX он принимает файловый дескриптор.

```
1 mov eax, 6 ; sys_close
2 mov ebx, [fd]
3 int 0x80
```

Запись в файл

Для записи данных в файл используется тот же системный вызов, что и для вывода данных на экран, только вместо стандартного потока вывода (stdout = 1), в регистр EBX нужно сохранить файловый дескриптор.

```
1 mov eax, 4   ; sys_write
2 mov ebx, [fd]
3 mov ecx, msg
4 mov edx, len
5 int 0x80
```

Все остальные параметры остаются прежними. В регистр ECX нужно поместить адрес той области памяти, в которой лежат нужные данные, а через регистр EDX передается длина строки в байтах.

Открытие файла

Ниже показан пример открытия файла и чтения данных в буфер. Число 5 в регистре EAX задает системный вызов **sys_open**. Через регистр ECX задается режим открытия файла. Значение 0 означает открытие только для чтения. Через регистр EDX задаются права доступа к файлу, но только если это новый файл. После выполнения системного вызова файловый дескриптор также помещается в EAX.

```
1 mov eax, 5       ; sys_open
2 mov ebx, filename
3 mov ecx, 0     ; O_RDONLY
4 mov edx, 0     ; Права доступа к файлу
5 int 0x80
6
7 mov [fd], eax
```

После открытия файла, помещаем файловый дескриптор в переменную.

Чтение из файла

Чтение из файла производится с помощью уже известного нам системного вызова sys_read. В отличие от считывания данных из терминала, в регистр EBX нужно передать значение файлового дескриптора из регистра либо из переменной, в которую мы сохранили его ранее.

```
1 mov eax, 3      ; sys_read
2 mov ebx, [fd]
3 mov ecx, buffer
4 mov edx, 10
5 int 0x80
```

Важно отметить, что данный системный вызов по результату выполнения возвращает количество успешно считанных байт через регистр EAX.

Вывод данных из файла на экран

Поскольку, после успешного считывания данных из потока, связанного с файлом, в регистре EAX хранится число прочитанных байт, последующий вывод на экран можно реализовать следующим способом

```
1 mov edx, eax ; Число байт для вывода на экран
2 mov eax, 4
3 mov ebx, 1
4 mov ecx, buffer
5 int 0x80
6
7 mov eax, 1 ; Выход из программы
8 mov ebx, 0
9 int 0x80
```

Чтение из файла

Рассмотрим еще один пример считывания данных из файла. В данном случае, мы хотим задавать имя файла при запуске программы.

```
user@pc:~$ ./a.out text.txt
```

Все переданные аргументы (известные нам по языку C argc и argv) хранятся в стеке

```
pop ebx ; argc
pop ebx ; argv[0]
pop ebx ; argv[1]
mov eax, 5 ; sys_open
mov ecx, 0 ; O_RDONLY = 0
mov edx, 0
int 0x80
```

Если при запуске была передана одна строка, то ее адрес находится в стеке на 3 позиции. После третьей операции POP, регистр EBX указывает на нужную строку с именем файла.

Перемещение по файлу

Рассмотрим пример перемещения по файлу. Число 19 в регистре EAX задает системный вызов **sys_lseek**. Регистр EBX, как и во всех подобных функциях, принимает файловый дескриптор. Значение регистра ECX задает число байт, на которые мы хотим сдвинуться по файлу.

```
1 mov eax, 19
2 mov ebx, [fd]
3 mov ecx, 15
4 mov edx, 0   ; SEEK_SET
5 int 0x80
```

Значение в регистре EDX задает один из трех параметров

1. **SEEK_SET**
2. **SEEK_CUR**
3. **SEEK_END**

Перемещение по файлу

Напомним, что системный вызов **sys_lseek** позволяет перемещаться по файлу, задавая отступ либо от начала файла **(SEEK_SET)**, либо от текущего положения файлового дескриптора **(SEEK_CUR)**, либо от конца файла **(SEEK_END)**.

Константа	Значение EDX
SEEK_SET	0
SEEK_CUR	1
SEEK_END	2

Допустим, нам нужно считать последние 10 байт файла, для этого переместим дескриптор

```
1 mov eax, 19
2 mov ebx, [fd]
3 mov ecx, -10
4 mov edx, 2   ; SEEK_END
5 int 0x80
```