

Информатика

Занятие 5

Грозов Владимир Андреевич

va_groz@mail.ru

Логические операции

Логические операции в языке C:

- `&&` – логическое «И»
- `||` – логическое «ИЛИ»
- `!` – логическое «НЕ»

a	b	a && b	a b	!a
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Поразрядные (побитовые) операции

Поразрядные операции в языке C:

- $\&$ – побитовое «И»
- $|$ – побитовое «ИЛИ»
- \sim – побитовое «НЕ»
- \wedge – исключающее «Или»
- \ll – побитовый сдвиг влево
- \gg – побитовый сдвиг вправо

Другие поразрядные операции:

- Стрелка Пирса (НЕ-ИЛИ)
- Штрих Шеффера (НЕ-И)

Поразрядные (побитовые) операции

Стрелка Пирса и штрих Шеффера:

a	b	a & b	a b	Стрелка Пирса $\sim(a b)$	Штрих Шеффера $\sim(a \& b)$
0	0	0	0	1	1
0	1	0	1	0	1
1	0	0	1	0	1
1	1	1	1	0	0

Поразрядные (побитовые) операции

Арифметические сдвиги:

- $A \ll B == A * 2^B$
- $A \gg B == A / 2^B$

Примеры арифметического сдвига:

- $(3 \ll 7) == (3 * 2^7) == 384$
- $(3072 \gg 10) == (3072 / 2^{10}) == 3$
- $(365 \gg 6) == (365 / 2^6) == 5$

Различия между логическими и поразрядными операциями в языке C

Логические операции:

- `15 && 10 == 1`
- `31 || 128 == 1`
- `63 || 0 == 1`
- `!209 == 0`

Поразрядные операции:

- `15 & 10 == 10`
- `31 | 128 == 159`
- `63 | 0 == 63`
- `~209 == 46` (для беззнаковых однобайтных чисел!)
- `202 ^ 75 == 129`

Работа с отдельными битами числа

Задание: Инвертировать пятый и шестой биты младшего байта числа X.

1 способ

```
srand(time(NULL));
int X = rand();
int X_56 = X << 25 >> 30; // А сдвиги можно было бы заменить на умножения в цикле
X -= (X_56 << 5);          // И эти сдвиги тоже
switch (X_56) {
    case 0:
        X_56 = 3;
        break;
    case 1:
        X_56 = 2;
        break;
    case 2:
        X_56 = 1;
        break;
    default:
        X_56 = 0;
        break;
}
X += X_56 << 5;
```

Работа с отдельными битами числа

Задание: Инвертировать пятый и шестой биты младшего байта числа X.

2 способ

```
srand(time(NULL));  
int X = rand();  
int X_56 = ~X << 25 >> 30;  
X -= (X & 96); //  $96_{10} = 01100000_2$   
X_56 <<= 5;  
X += X_56;
```


Работа с отдельными битами числа

Задание: Инвертировать пятый и шестой биты младшего байта числа X.

3 способ

```
srand(time(NULL));
```

```
int X = rand();
```

```
X ^= 96;    //  $96_{10} = 01100000_2$ 
```

Работа с отдельными битами числа

Задание: выполнить циклический сдвиг числа x вправо на 2.

1 способ

```
 srand(time(NULL));  
 int x = rand(), y = 0;  
 for (int i = 0; i < 2; ++i)  
 {  
     y = x % 2;  
     x = (x / 2) + (y * 2 * 1024 * 1024 * 1024);  
 }
```

Работа с отдельными битами числа

Задание: выполнить циклический сдвиг числа x вправо на 2.

2 способ

```
 srand(time(NULL));
 int x = rand(), y = 0;
 for (int i = 0; i < 2; ++i)
 {
     y = x & 1;           //  $1_{10} = 00000001_2$ 
     x = (x >> 1) + (y << 31);
 }
```

Работа с отдельными битами числа

Задание: выполнить циклический сдвиг числа x вправо на 2.

3 способ

```
srand(time(NULL));  
int x = rand();  
int y = x % 4; // А лучше так: y = x & 3;  
y <<= 30;  
x >>= 2;  
x += y;
```

Задание 1

Задача:

Напишите программу на языке C, которая инвертирует второй по старшинству байт случайного числа X типа `int`.

Задание 2

Задача:

Напишите программу на языке C, которая меняет местами нулевой и третий, а также первый и второй байты случайного числа X типа `int`.

Задание 3

Задача:

Напишите программу на языке C, которая обнуляет биты случайного числа X типа `int` с номерами 0, 5, 12 и 14.

Задание 4

Задача:

Напишите программу на языке C, которая заносит в биты случайного числа X типа `int` с номерами 3, 8 и 10 значение 1.

Задание 5

Задача:

Напишите программу на языке C, которая выполняет циклический сдвиг влево на 3 первого байта случайного числа X типа `int` (нумерация байтов начинается с нуля!).

Задание 6

Задача:

Напишите программу на языке C, которая выполняет проверку значения седьмого бита случайного числа `X` типа `int` (нумерация битов начинается с нуля!). Если этот бит равен 0, инвертировать значение 12-го бита.

Задание 7

Задача:

Напишите программу на языке C, которая получает случайное число X типа `int`. В этом числе седьмой бит переместить в одиннадцатый, а третий и шестой поменять местами. Нумерация битов начинается с нуля.

Ассемблер. NASM

```
section .data
str1    db  'Here is string 1', 0xA
str1_len    equ $ - str1

pi      dw  0x123d
ksi     dd  0x12345678
ksi2    dd  'acde'

; -----
section .bss
mem     resb  12800

; -----
section .text
global _start
_start:
    mov eax, 4
    mov ebx, 1
    mov ecx, str1
    mov edx, str1_len
    int 80h

    mov ecx, str1_len

.loop:
    mov esi, ecx
    mov al, byte [str1+esi]
    mov byte [mem+esi], al
    loop .loop

    mov eax, 4
    mov ebx, 1
    mov ecx, mem
    mov edx, str1_len
    int 80h

    push 1234abcdh
    call print_hex

    mov eax, 1
    mov ebx, 0
    int 80h
```

Установка NASM

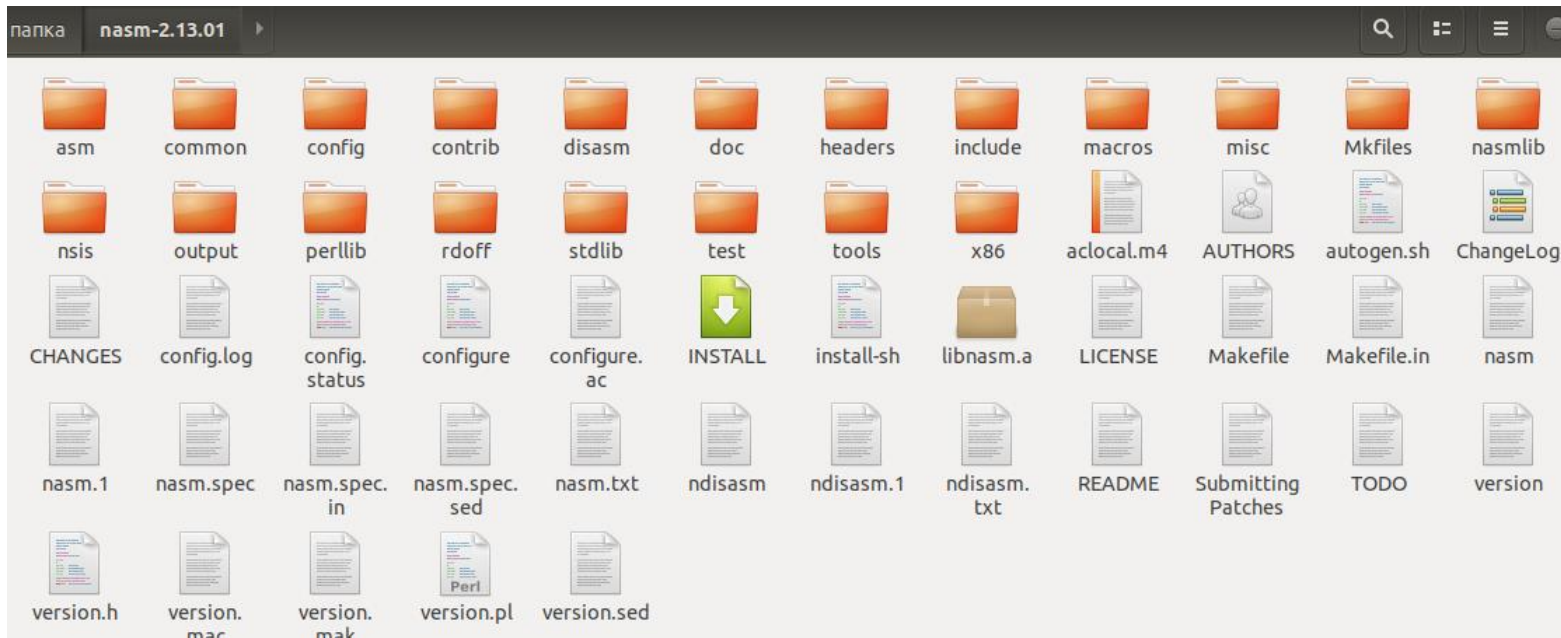
Установка NASM:

- **Проверка наличия NASM в ОС:** `nasm -h` (или просто `nasm`)
- **Онлайн-компиляторы NASM (и не только):**
 - myCompiler: https://www.mycompiler.io/new/asm-x86_64
 - OneCompiler: <https://onecompiler.com/assembly/3ymh46gj2>
- **1 способ установки NASM:**
 - Терминал -> `apt install nasm`
 - Ввод пароля!

Установка NASM

Установка NASM:

- **2 способ установки NASM:**
 1. Загрузка архива, содержащего NASM
 2. Разархивировать (например, в папку `nasm_cat`), открыть файл `INSTALL`



Установка NASM

Установка NASM:

3. Терминал -> `cd ./nasm_cat`
4. `sh autogen.sh` (может не понадобиться)
5. `sh configure`
6. `make`
 1. Или: `make everything` (более полное построение)
 2. Или: `make strip` (игнорирование необязательных данных)
7. Переход в root (команда `su` или `sudo`)
8. Переход обратно в папку с `nasm`
9. `make install`
10. NASM установлен!

Установка NASM

```
root@vladimir-Vostro-3490: ~  
Файл Правка Вид Поиск Терминал Справка  
/usr/bin/install: невозможно удалить '/usr/local/bin/nasm': Отказано в доступе  
Makefile:337: recipe for target 'install' failed  
make: *** [install] Error 1  
vladimir@vladimir-Vostro-3490:~/nasm-2.13.01$ sudo  
usage: sudo -h | -K | -k | -V  
usage: sudo -v [-AknS] [-g group] [-h host] [-p prompt] [-u user]  
usage: sudo -l [-AknS] [-g group] [-h host] [-p prompt] [-U user] [-u user]  
[command]  
usage: sudo [-AbEHknPS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p  
prompt] [-T timeout] [-u user] [VAR=value] [-i|-s] [<command>]  
usage: sudo -e [-AknS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p  
prompt] [-T timeout] [-u user] file ...  
vladimir@vladimir-Vostro-3490:~/nasm-2.13.01$ sudo su  
[sudo] пароль для vladimir:  
root@vladimir-Vostro-3490:/home/vladimir/nasm-2.13.01# ~  
bash: /root: Это каталог  
root@vladimir-Vostro-3490:/home/vladimir/nasm-2.13.01# ..  
..: команда не найдена  
root@vladimir-Vostro-3490:/home/vladimir/nasm-2.13.01# /~  
bash: /~: Нет такого файла или каталога  
root@vladimir-Vostro-3490:/home/vladimir/nasm-2.13.01# ~  
bash: /root: Это каталог  
root@vladimir-Vostro-3490:/home/vladimir/nasm-2.13.01# cd ~  
root@vladimir-Vostro-3490:~# make install
```


Написание программ NASM

Написание программ:

- Любой текстовый редактор (стандартный или любой другой)
- Расширение (суффикс) файла - *.asm

Компиляция NASM

Компиляция:

- **Проходит в 2 этапа:**

- **Ассемблирование**

- На выходе – объектный файл file.o

- **Сборка (компановка)**

- Компановка выполняется из одного или нескольких объектных модулей. На выходе – исполняемый файл программы (например, prog). В Linux часто используется стандартный компановщик ld.

Ассемблирование

Ассемблирование:

`nasm -f <format> <имя_файла.asm> [-o <объектный_файл>]`

- **Формат выходных файлов:**
 - elf
 - bin
 - obj
 - coff
 - ... (их много)

Часто используется формат elf

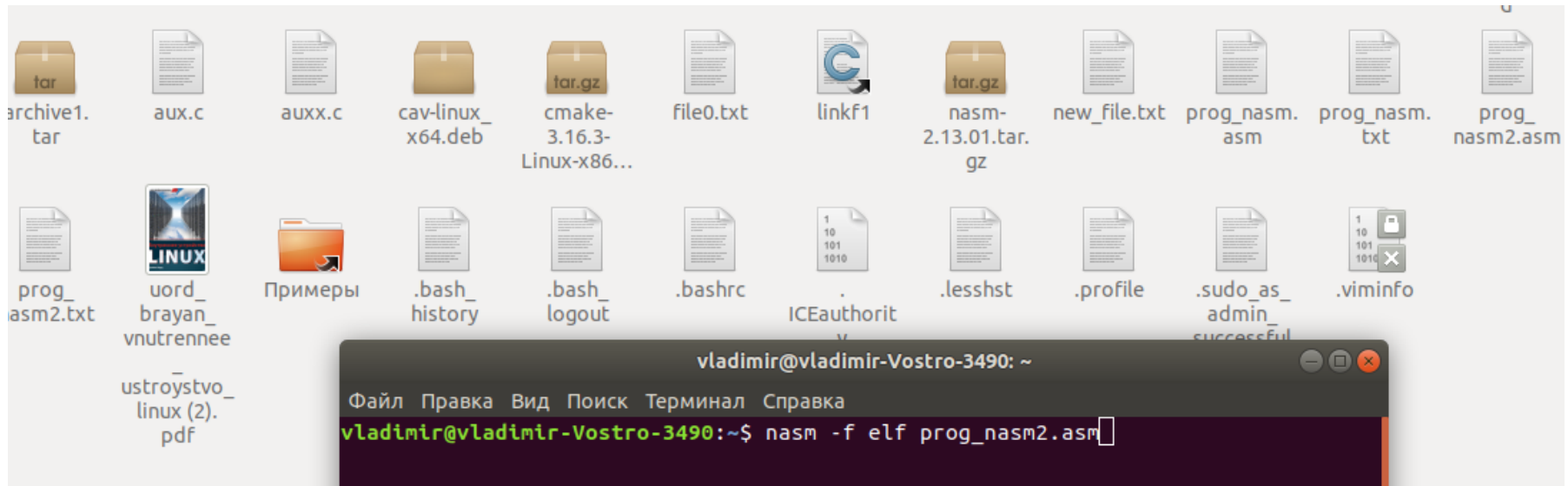
Ассемблирование

Пример: `nasm -f elf prog_nasm.asm`

Результат: `prog_nasm.o`

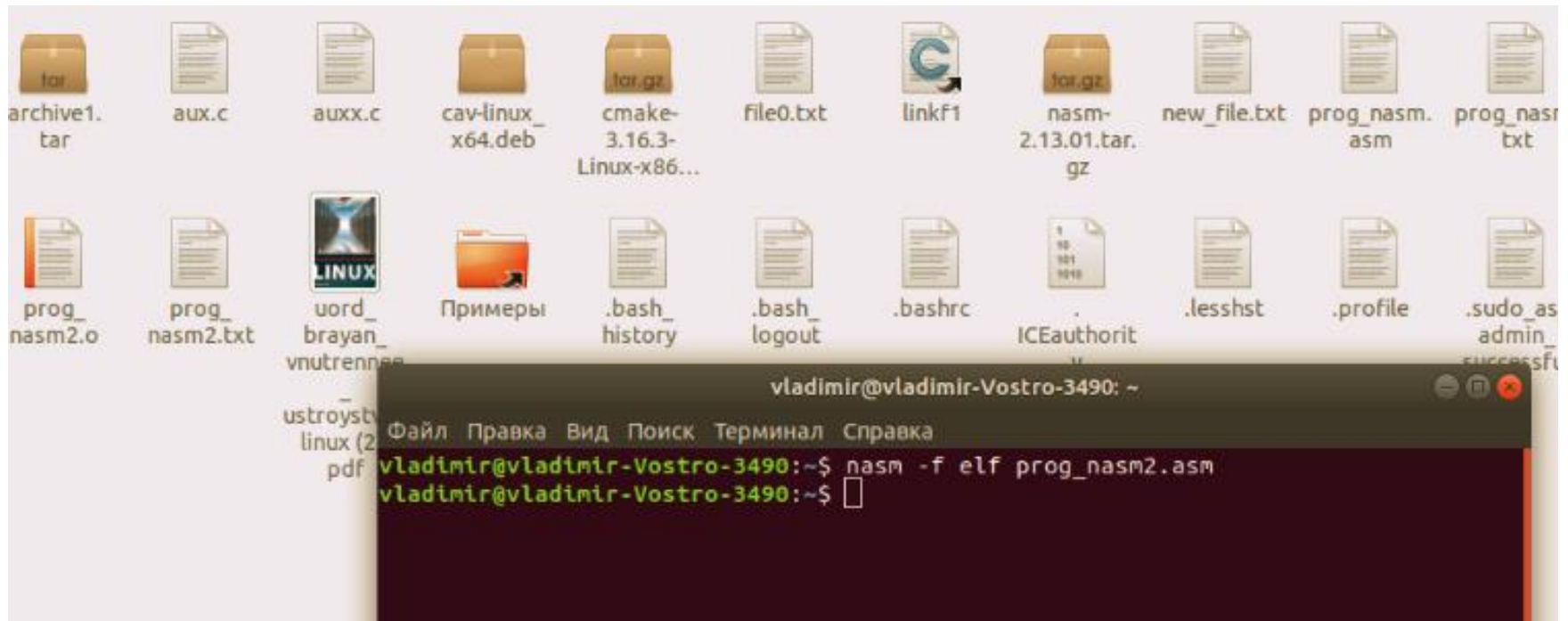
Или: `nasm -f elf prog_nasm.asm -o abc.o`

Результат: `abc.o`



Ассемблирование

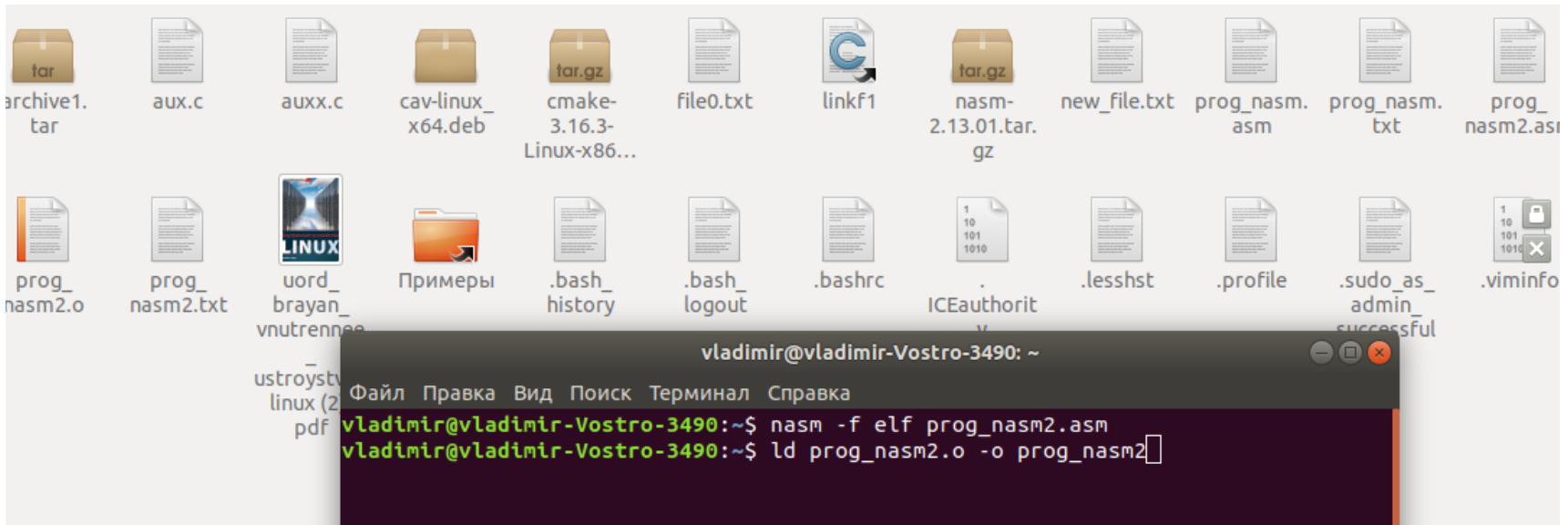
`nasm -f elf prog_nasm2.asm`



Компановка

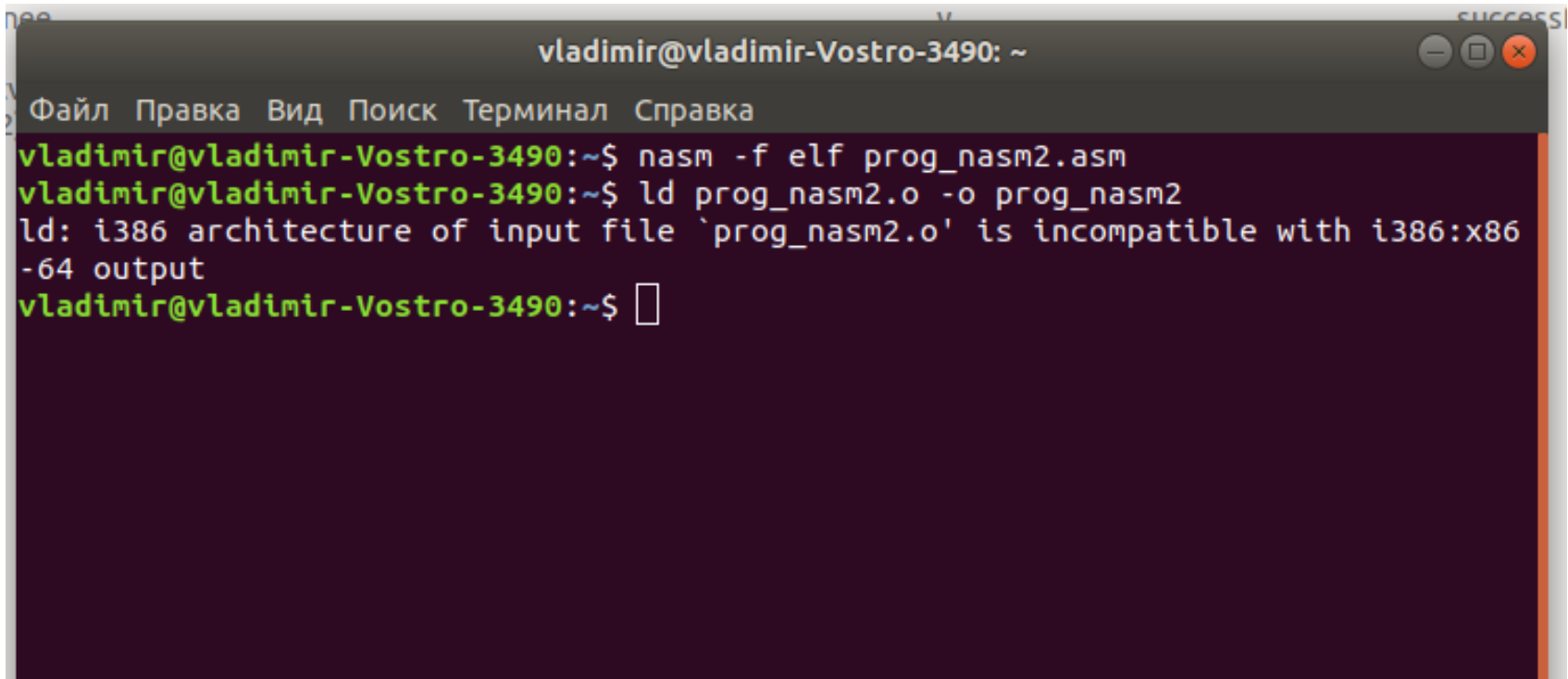
Компановщик – ld ([эль-дэ]).

- ld prog_nasm2.o -o prog_nasm2



Компановка

Используемая ОС – Linux Ubuntu x64, поэтому выведено сообщение об ошибке

A screenshot of a terminal window titled 'vladimir@vladimir-Vostro-3490: ~'. The window has a menu bar with 'Файл', 'Правка', 'Вид', 'Поиск', 'Терминал', and 'Справка'. The terminal shows the following commands and output:

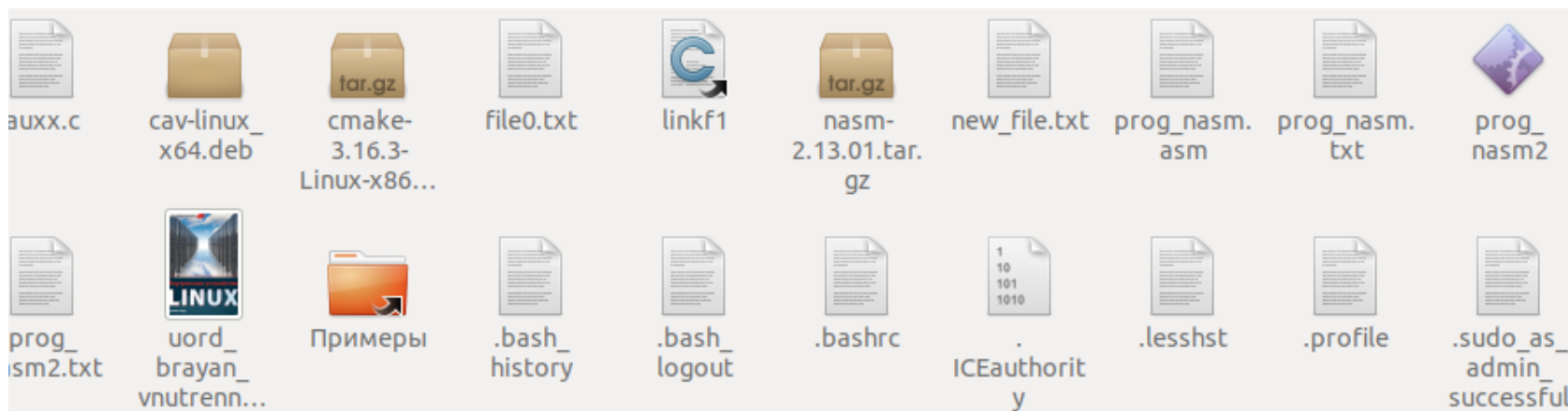
```
vladimir@vladimir-Vostro-3490:~$ nasm -f elf prog_nasm2.asm
vladimir@vladimir-Vostro-3490:~$ ld prog_nasm2.o -o prog_nasm2
ld: i386 architecture of input file `prog_nasm2.o' is incompatible with i386:x86-64 output
vladimir@vladimir-Vostro-3490:~$
```

The error message indicates an incompatibility between the i386 architecture of the input file and the i386:x86-64 output.

Компановка

Для 64-битных ОС:

- `ld -m elf_i386 prog_nasm2.o -o prog_nasm2`

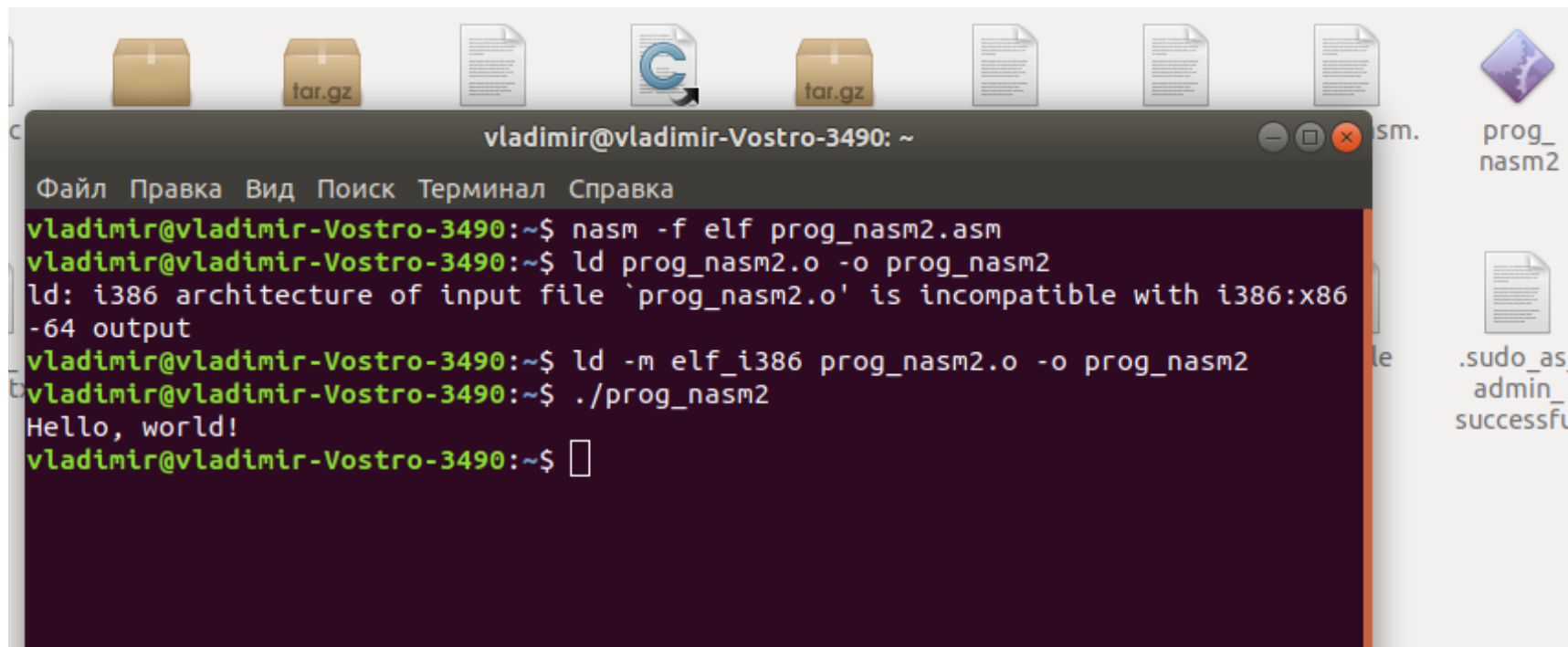


```
vladimir@vladimir-Vostro-3490: ~  
Файл Правка Вид Поиск Терминал Справка  
vladimir@vladimir-Vostro-3490:~$ nasm -f elf prog_nasm2.asm  
vladimir@vladimir-Vostro-3490:~$ ld prog_nasm2.o -o prog_nasm2  
ld: i386 architecture of input file `prog_nasm2.o' is incompatible with i386:x86-64 output  
vladimir@vladimir-Vostro-3490:~$ ld -m elf_i386 prog_nasm2.o -o prog_nasm2  
vladimir@vladimir-Vostro-3490:~$
```


Запуск программ NASM

Запуск:

`./prog_nasm2`



The screenshot shows a terminal window titled "vladimir@vladimir-Vostro-3490: ~". The window has a menu bar with "Файл", "Правка", "Вид", "Поиск", "Терминал", and "Справка". The terminal output is as follows:

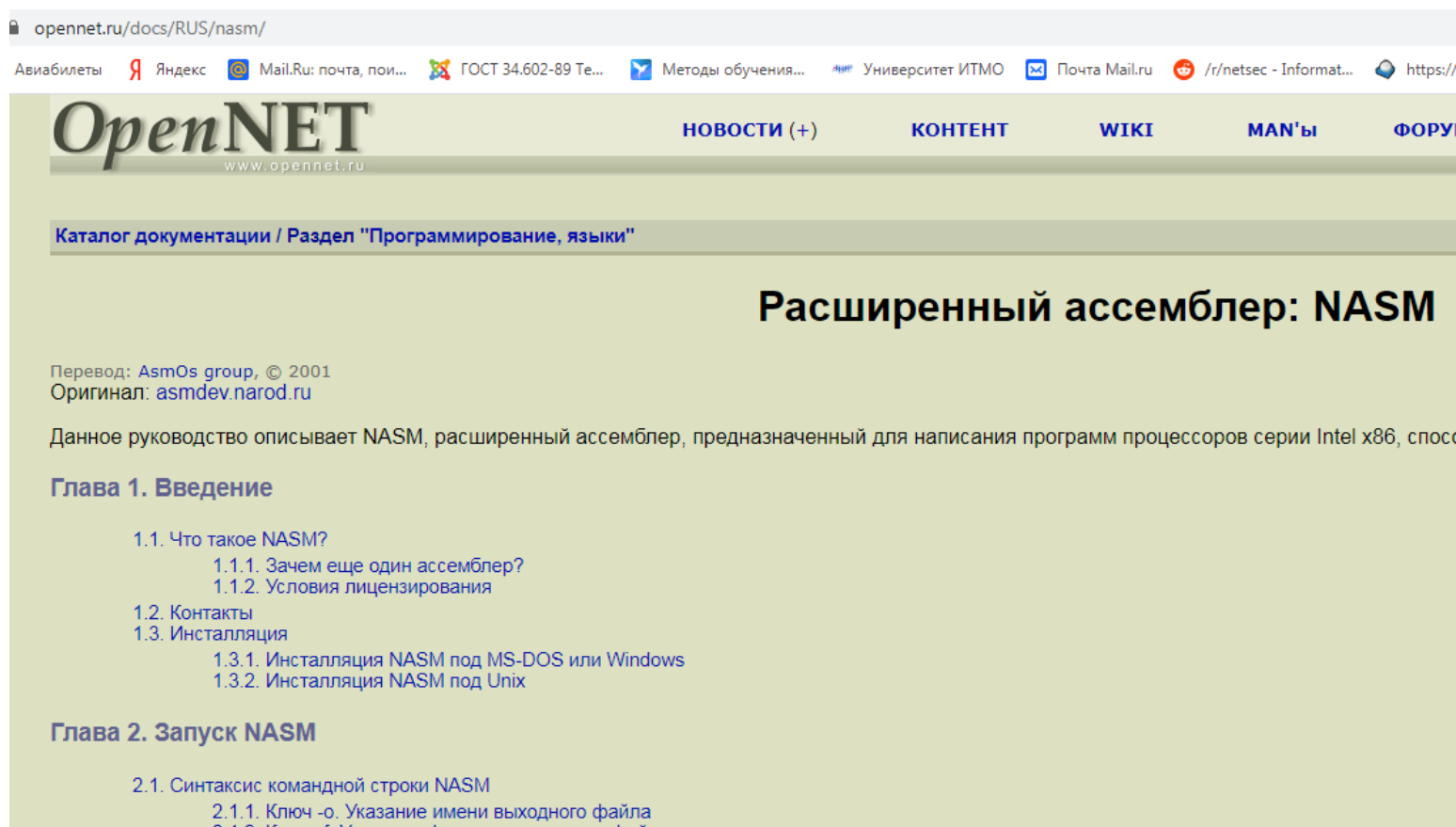
```
vladimir@vladimir-Vostro-3490:~$ nasm -f elf prog_nasm2.asm
vladimir@vladimir-Vostro-3490:~$ ld prog_nasm2.o -o prog_nasm2
ld: i386 architecture of input file `prog_nasm2.o' is incompatible with i386:x86-64 output
vladimir@vladimir-Vostro-3490:~$ ld -m elf_i386 prog_nasm2.o -o prog_nasm2
vladimir@vladimir-Vostro-3490:~$ ./prog_nasm2
Hello, world!
vladimir@vladimir-Vostro-3490:~$
```

The background of the terminal window shows a desktop environment with several icons: two tar.gz files, a document icon, a blue 'C' icon, and a purple icon labeled "prog_nasm2". To the right of the terminal window, there is a file icon labeled ".sudo_admin_successfu".

Рекомендуемая литература

1. Расширенный ассемблер: NASM

<https://www.opennet.ru/docs/RUS/nasm/>



The screenshot shows a web browser window with the address bar displaying `opennet.ru/docs/RUS/nasm/`. The browser's toolbar includes various icons and links such as "Авиабилеты", "Яндекс", "Mail.Ru: почта, пои...", "ГОСТ 34.602-89 Те...", "Методы обучения...", "Университет ИТМО", "Почта Mail.ru", and "/r/netsec - Informat...". The website header features the "OpenNET" logo and navigation links: "НОВОСТИ (+)", "КОНТЕНТ", "WIKI", "MAN'ы", and "ФОРУМ". Below the header, a breadcrumb trail reads "Каталог документации / Раздел 'Программирование, языки'". The main heading is "Расширенный ассемблер: NASM". Below this, it states "Перевод: AsmOs group, © 2001" and "Оригинал: asmdev.narod.ru". A paragraph follows: "Данное руководство описывает NASM, расширенный ассемблер, предназначенный для написания программ процессоров серии Intel x86, спосо...". The table of contents lists "Глава 1. Введение" with sub-items: "1.1. Что такое NASM?" (including "1.1.1. Зачем еще один ассемблер?" and "1.1.2. Условия лицензирования"), "1.2. Контакты", and "1.3. Установка" (including "1.3.1. Установка NASM под MS-DOS или Windows" and "1.3.2. Установка NASM под Unix"). "Глава 2. Запуск NASM" includes "2.1. Синтаксис командной строки NASM" (including "2.1.1. Ключ -o. Указание имени выходного файла" and "2.1.2. Ключ -f. Указание формата выходного файла").

2. А.В. Столяров. Программирование на языке ассемблера NASM для OS UNIX. 2011



Пример кода NASM

```
section .data
str1    db  'Here is string 1', 0xA
str1_len equ $ - str1
pi       dw  0x123d
ksi      dd  0x12345678
ksi2     dd  'acde'

; -----
section .bss
mem      resb  12800

; -----
section .text
global _start
_start:
    mov eax, 4
    mov ebx, 1
    mov ecx, str1
    mov edx, str1_len
    int 80h

    mov ecx, str1_len
.loop:
    mov esi, ecx
    mov al, byte [str1+esi]
    mov byte [mem+esi], al
    loop .loop

    mov eax, 4
    mov ebx, 1
    mov ecx, mem
    mov edx, str1_len
    int 80h

    push 1234abc1h
    call print_hex

    mov eax, 1
    mov ebx, 0
    int 80h
```