

# Информатика

## Занятие 5

Грозов Владимир Андреевич

[va\\_groz@mail.ru](mailto:va_groz@mail.ru)

[vk.com/vl\\_grozov](https://vk.com/vl_grozov)

# 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»
- `~` – побитовое «НЕ»
- `^` – исключающее «Или»
- `<<` – побитовый сдвиг влево
- `>>` – 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<br>$\sim(a   b)$ | Штрих Шеффера<br>$\sim(a \& b)$ |
|---|---|-------|-------|--------------------------------|---------------------------------|
| 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:

- $15 \ \& \ 10 \ == \ 10$
- $31 \ | \ 128 \ == \ 159$
- $63 \ | \ 0 \ == \ 63$
- $\sim 209 \ == \ 46$  (для беззнаковых однобайтных чисел!)
- $202 \ \wedge \ 75 \ == \ 129$

# 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 << 26 >> 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;
```

# 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 << 26 >> 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      dq  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

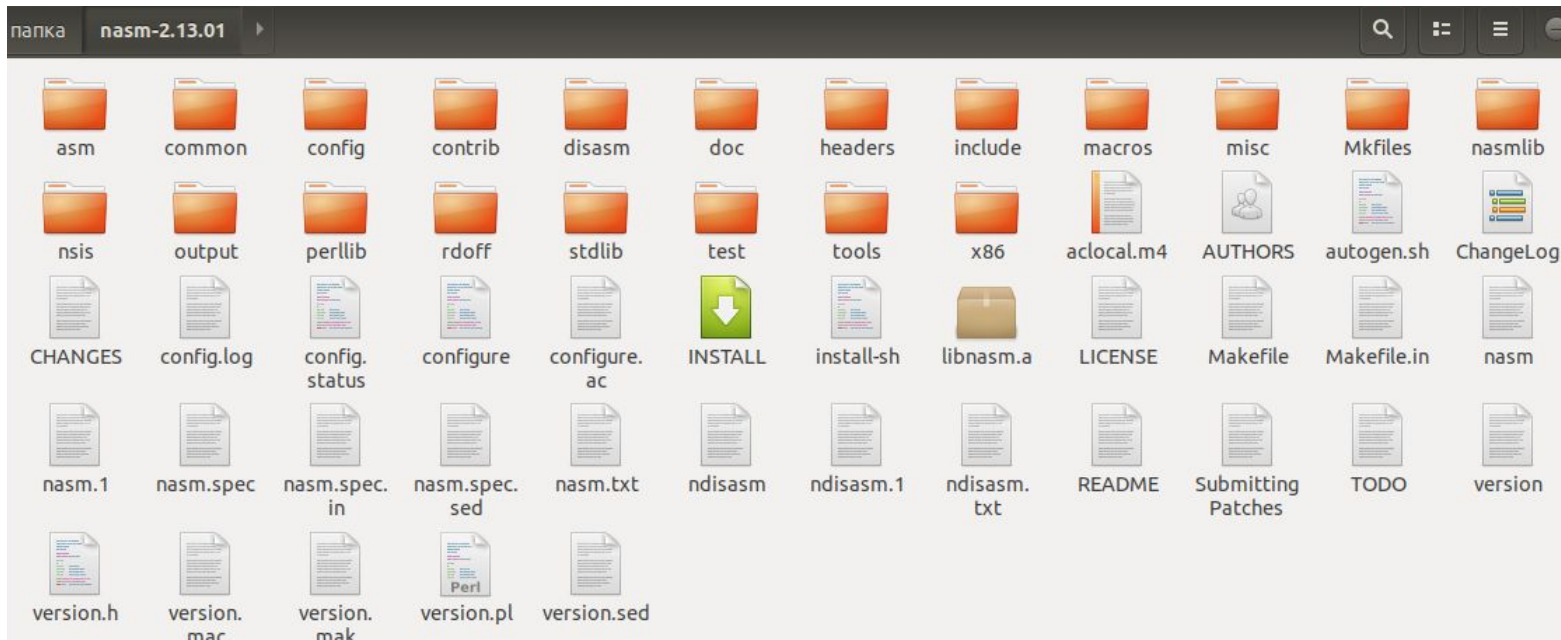
## Установка NASM:

- **Проверка наличия NASM в ОС:** `nasm -h`  
(или просто `nasm`)
- **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

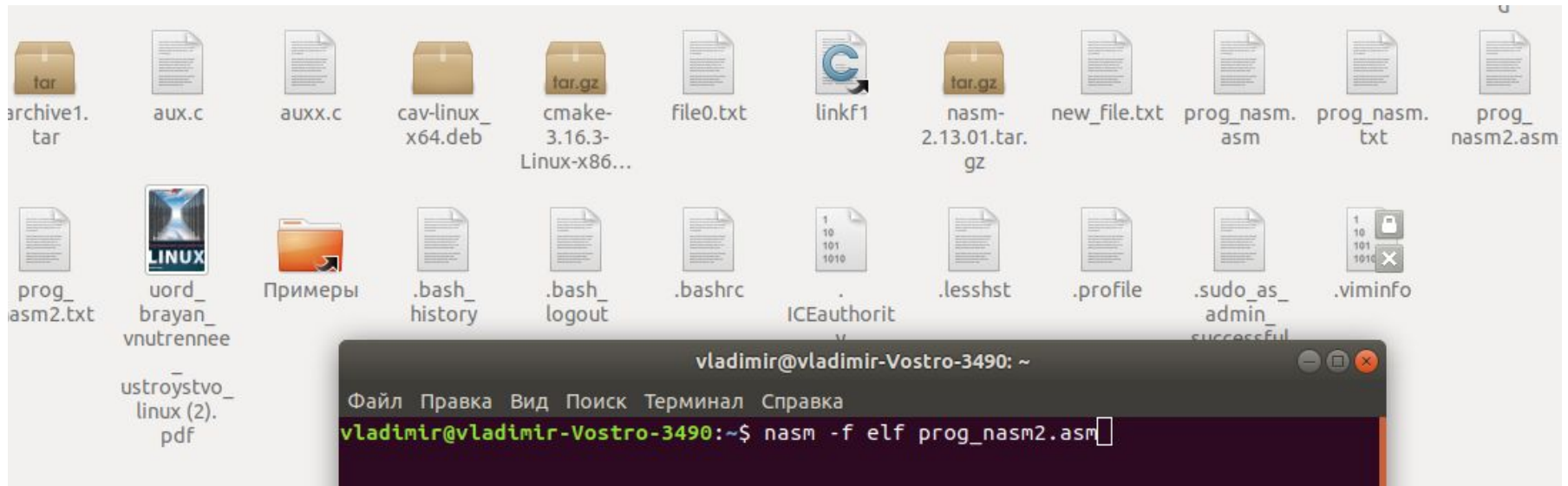
# Ассемблирование

**Пример:** `nasm -f elf prog_nasm.asm`

Результат: `prog_nasm.o`

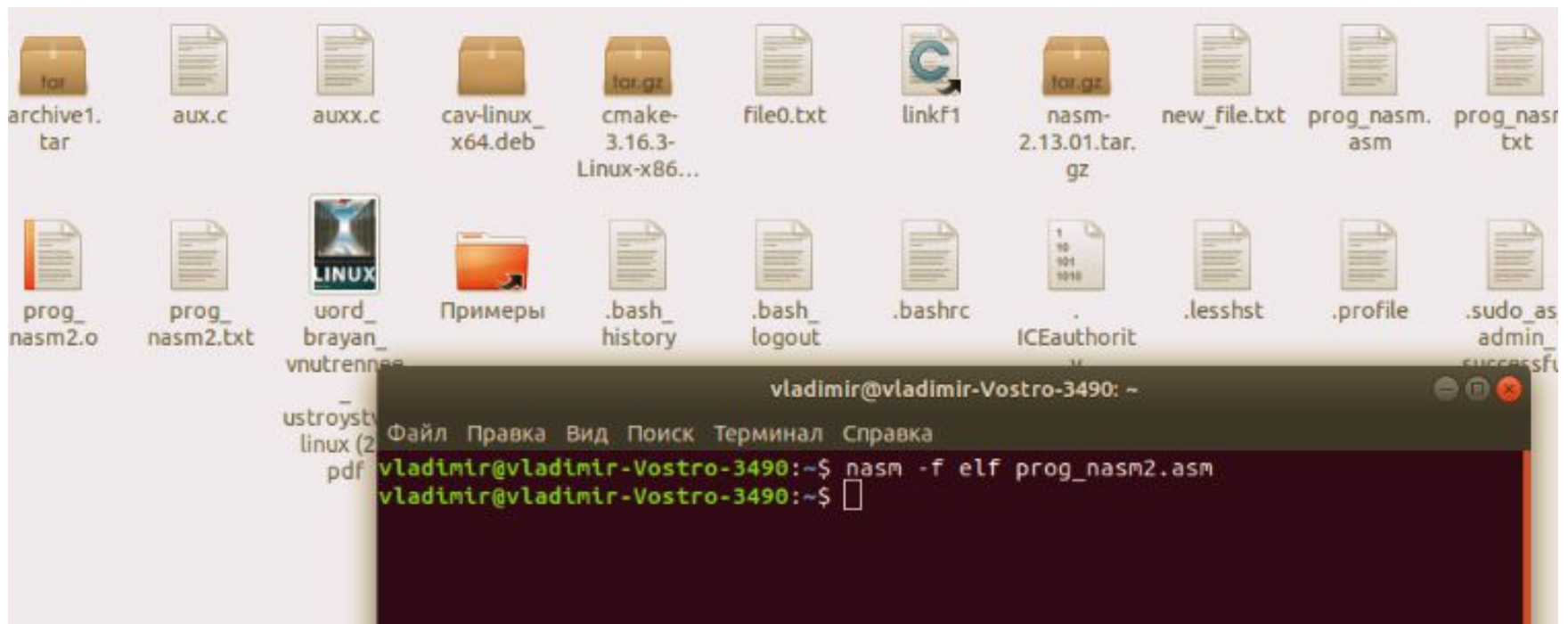
**Или:** `nasm -f elf prog_nasm.asm -o abc.o`

Результат: `abc.o`



# Ассемблирование

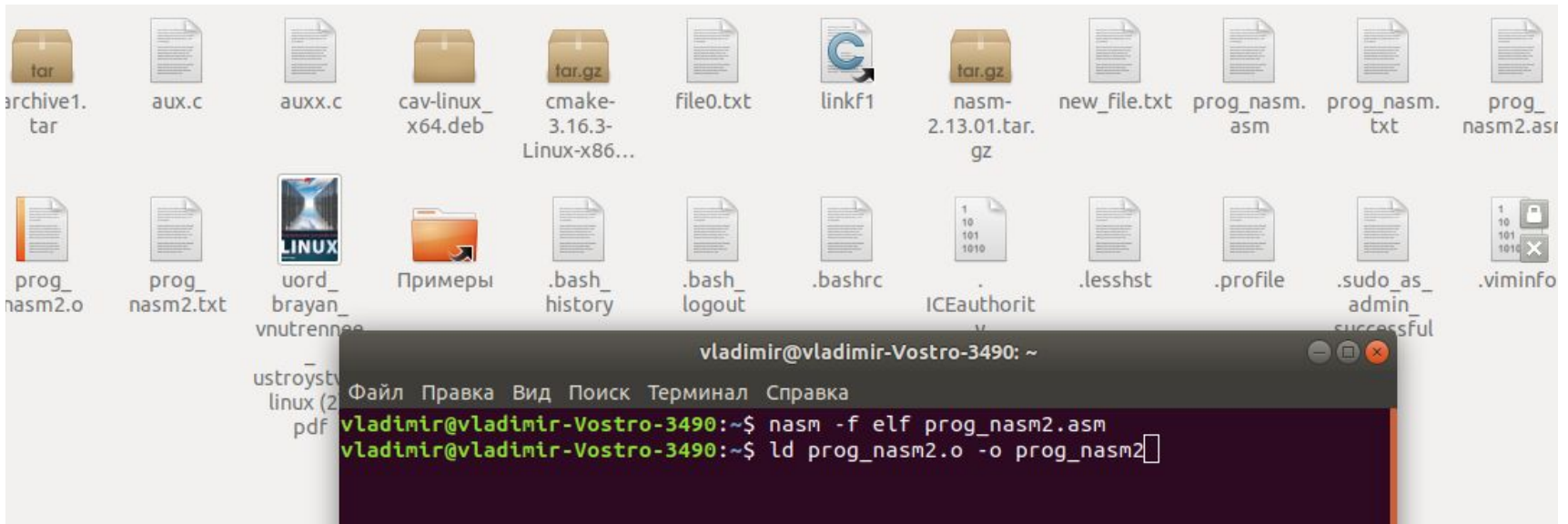
```
nasm -f elf prog_nasm2.asm
```



# Компановка

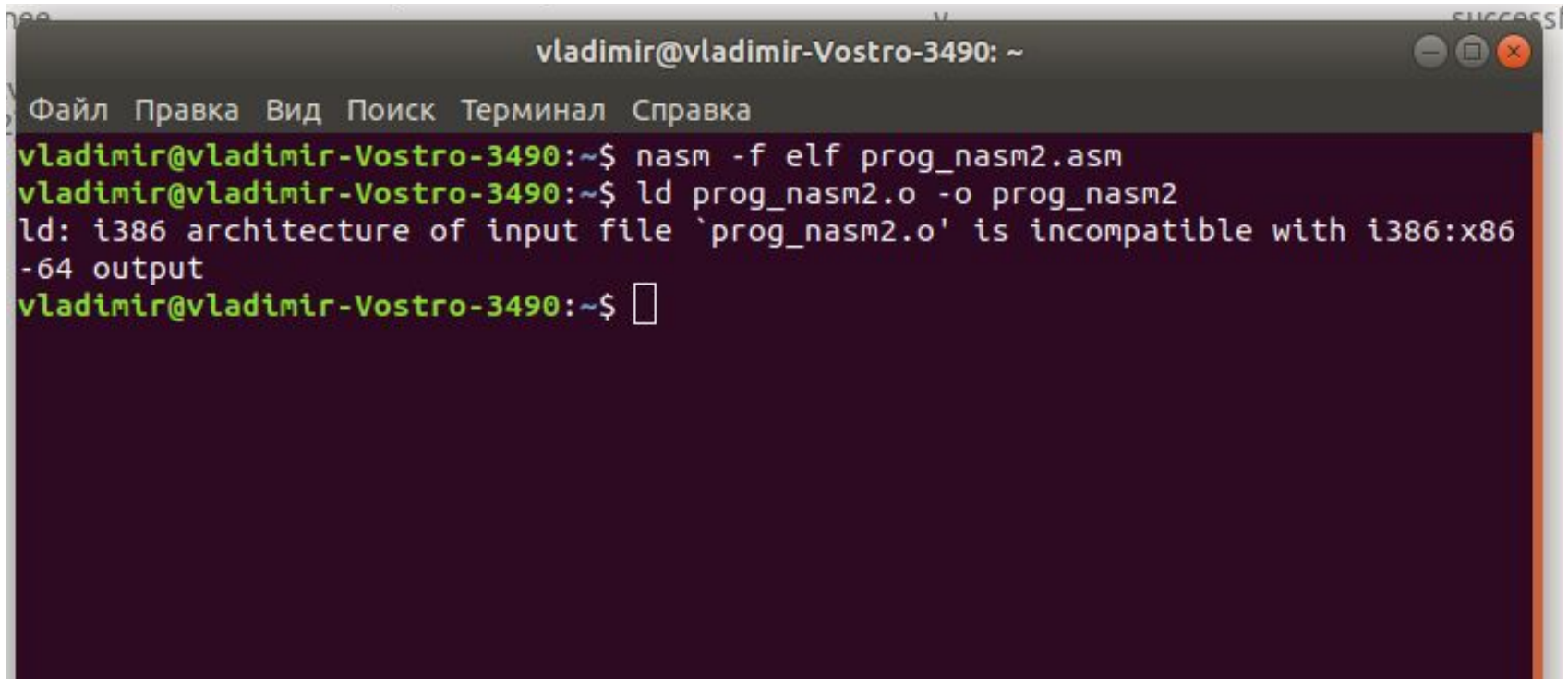
Компановщик – ld ([эль-дэ]).

- ld prog\_nasm2.o -o prog\_nasm2



# Компановка

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

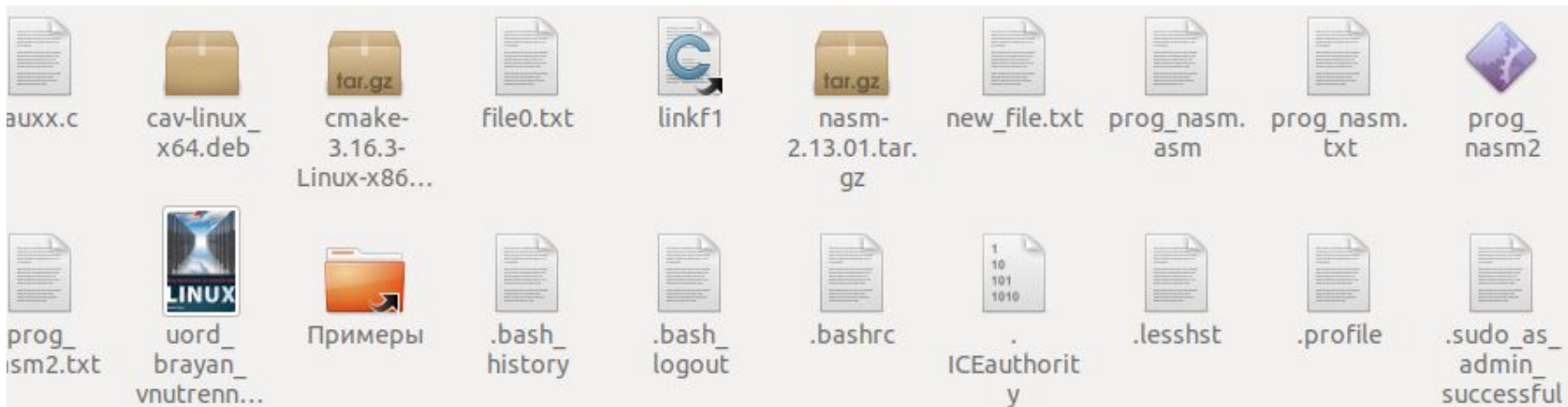
A terminal window titled 'vladimir@vladimir-Vostro-3490: ~' with a menu bar containing 'Файл', 'Правка', 'Вид', 'Поиск', 'Терминал', and 'Справка'. The terminal shows the following commands and output:

```
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:~$
```

# Компановка

Для 64-БИТНЫХ ОС:

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



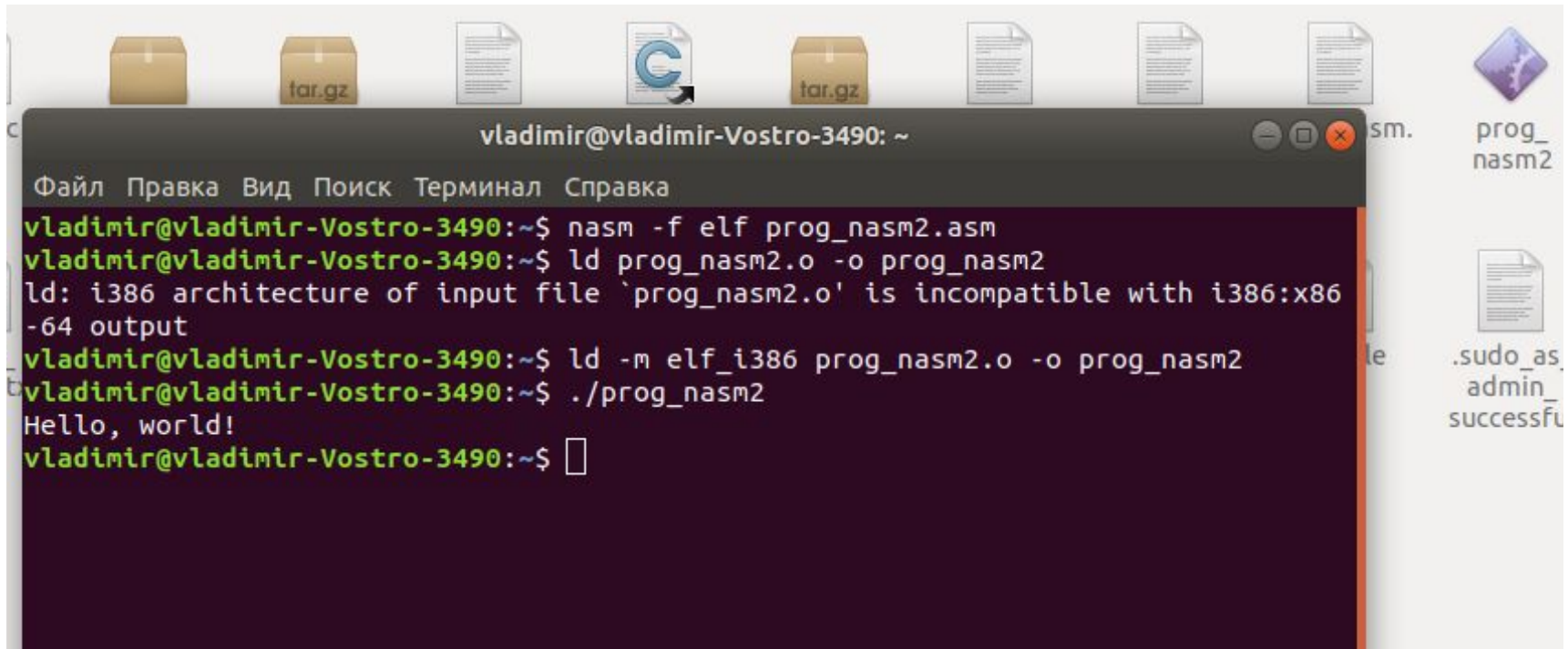
```
vladimir@vladimir-Vostro-3490: ~  
Файл Правка Вид Поиск Терминал Справка  
vladimir@vladimir-Vostro-3490:~$ nasm -f elf prog_nasm2.asm  
vladimir@vladimir-Vostro-3490:~$ ld prog_nasm2.o -o prog_nasm2  
ld: i386 architecture of input file `prog_nasm2.o' is incompatible with i386:x86-64 output  
vladimir@vladimir-Vostro-3490:~$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

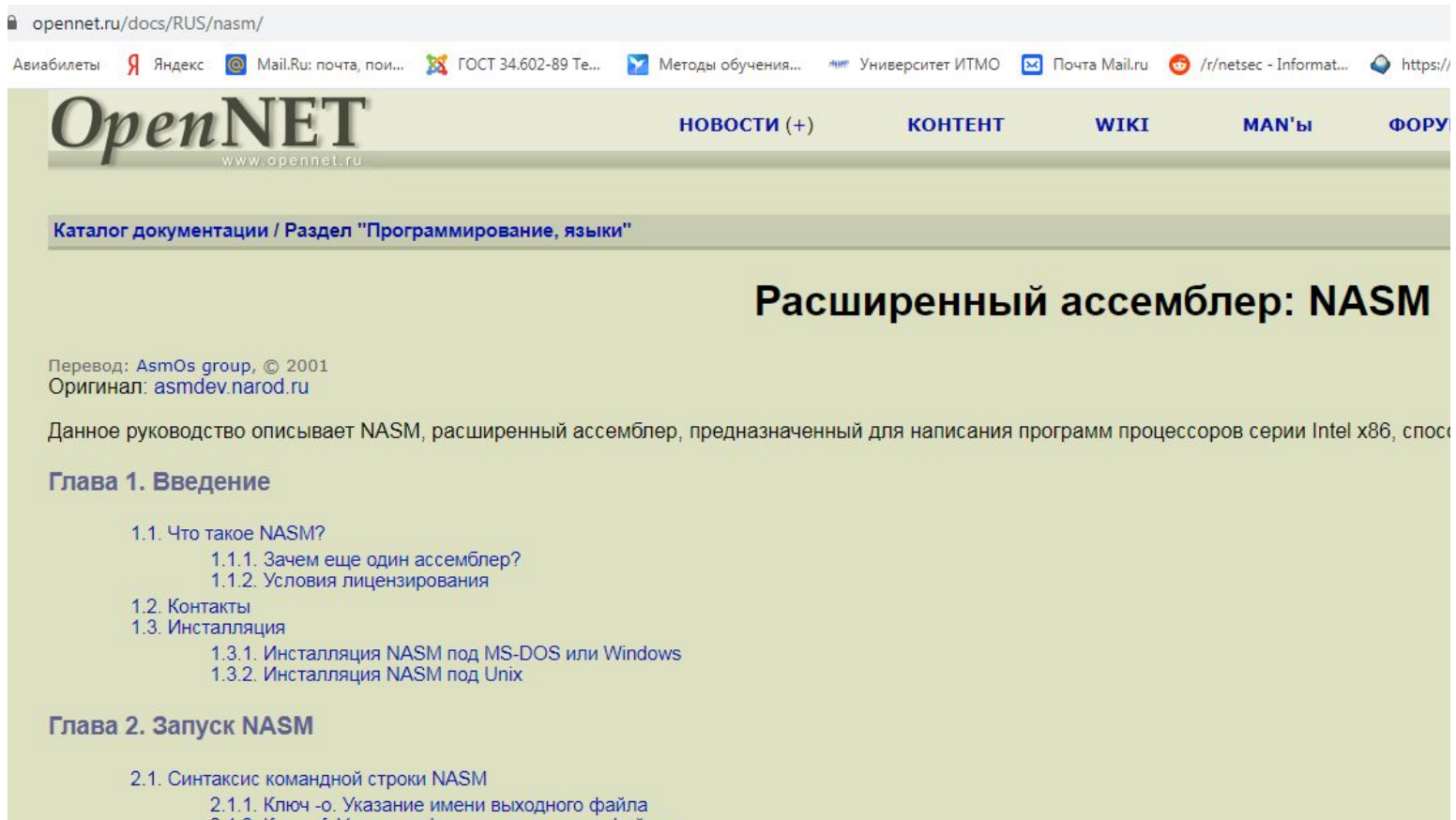
```
Файл Правка Вид Поиск Терминал Справка
vladimir@vladimir-Vostro-3490:~$ nasm -f elf prog_nasm2.asm
vladimir@vladimir-Vostro-3490:~$ ld prog_nasm2.o -o prog_nasm2
ld: i386 architecture of input file `prog_nasm2.o' is incompatible with i386:x86-64 output
vladimir@vladimir-Vostro-3490:~$ ld -m elf_i386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

The terminal window is overlaid on a desktop environment. The desktop background is light gray. In the top panel, there are several icons: a folder, a tar.gz file, a document, a blue 'C' logo, another tar.gz file, another document, another document, and another document. On the right side of the desktop, there are icons for "prog\_nasm2" (a purple diamond) and ".sudo\_admin\_successfu" (a document).

# Рекомендуемая литература

## 1. Расширенный ассемблер: NASM

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



The screenshot shows a web browser window with the address bar containing `opennet.ru/docs/RUS/nasm/`. The browser's taskbar includes icons for Aviaбилеты, Яндекс, Mail.Ru, GOCT 34.602-89, Методы обучения..., Университет ИТМО, Почта Mail.ru, /r/netsec - Informat..., and https://. 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, спос...". The table of contents includes:

- Глава 1. Введение**
  - 1.1. Что такое NASM?
    - 1.1.1. Зачем еще один ассемблер?
    - 1.1.2. Условия лицензирования
  - 1.2. Контакты
  - 1.3. Установка
    - 1.3.1. Установка NASM под MS-DOS или Windows
    - 1.3.2. Установка NASM под Unix
- Глава 2. Запуск NASM**
  - 2.1. Синтаксис командной строки NASM
    - 2.1.1. Ключ `-o`. Указание имени выходного файла
    - 2.1.2. Ключ `-f`. Указание формата выходного файла



# Пример кода NASM

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

pi      dq  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
```