

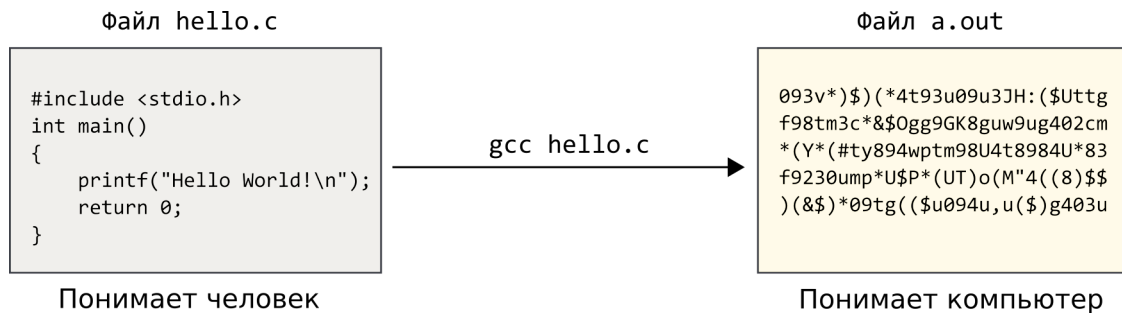
Семинар #1: Работа с командной строкой. Основы С.

Часть 1: Работа с командной строкой:

Основные команды терминала Linux:

<code>pwd</code>	напечатать имя текущей директории
<code>ls</code>	напечатать все файлы и папки текущей директории
<code>cd имя_папки</code>	перейти в соответствующую папку
	например: <code>cd /home-local/student</code>
<code>cd ..</code>	перейти в папку, содержащую данную
<code>mkdir имя_новой_папки</code>	создать новую папку
<code>gcc имя_файла_исходного_кода</code>	скомпилировать программу и создать исполняемый файл <code>a.out</code>
<code>./a.out</code>	запустить файл <code>a.out</code> в текущей директории

Если вы используете Windows, то вместо команды `ls` нужно будет использовать команду `dir`, а запустить программу нужно будет командой `a.exe` вместо `./a.out`.



Горячие клавиши:

<code>Tab</code>	автозаполнение
<code>2 раза Tab</code>	показать возможные варианты
<code>стрелка вверх</code>	перейти к предыдущей команде
<code>Ctrl-C</code>	выход из программы, например той, которая зависла в бесконечном цикле
<code>Ctrl-R</code>	поиск по всем предыдущим командам

Задание на работу с командной строкой:

1. Откройте терминал и узнайте в какой папке вы находитесь. Для этого напечатайте `pwd` и нажмите `Enter`.
2. Перейдите в папку `/home-local/student`. Для этого введите команду:

```
cd /home-local/student
```

3. С помощью команды `pwd` проверьте, что вы действительно находитесь в нужной папке.
4. С помощью команды `ls` просмотрите всё содержимое папки `/home-local/student`. Для этого введите `ls` и нажмите `Enter`.
5. Создайте вашу папку, в которой вы будете работать в течении семестра. Используйте команду:

```
mkdir имя_папки
```

За место `имя_папки` подставьте название вашей папки. Желательно, чтобы название содержало только латинские символы без пробелов.

6. С помощью команды `ls` убедитесь, что ваша папка создавалась.

7. Перейдите в вашу созданную папку командой `cd имя_папки`.
8. Перейдите в эту папку с помощью файлового менеджера(проводника) и создайте там файл `hello.c`. Файл обязан оканчиваться на `.c`.
9. С помощью обычного текстового редактора (например, gedit или Sublime Text) напишите в файле `hello.c` текст программы *HelloWorld*:

```
#include <stdio.h>
int main()
{
    printf("Hello World\n");
}
```

10. В терминале проверьте, что этот файл существует, используя команду `ls`.
11. Скомпилируйте этот файл следующей командой:

```
gcc hello.c
```

После этого в папке создастся новый файл по имени `a.out`.

12. Запустите исполняемый файл `a.out` напечатав полный путь до этого файла:

```
/home-local/student/ваша_папка/a.out
```

13. Точка в имени файлового пути является сокращением для текущей папки. То есть в данном случае точка является сокращением для `/home-local/student/ваша_папка`. Поэтому команду для запуска файла `a.out` можно сократить до:

```
./a.out
```

14. Можно объединить команды компиляции и запуска:

```
gcc hello.c && ./a.out
```

Измените программу так, чтобы она печатала `Hello MIPТ!`, скомпилируйте и запустите программу.

Примечание: каждый раз вводить эту команду не нужно, можно просто нажать клавишу вверх, чтобы исполнить предыдущие команды.

15. Перейдите в папку `code/1int`, используя `cd`.
16. Скомпилируйте и запустите программу `01print_int.c` с помощью команды

```
gcc 01print_int.c && ./a.out
```

Эта программа должна напечатать на экран:

```
My name is Alex
I am 10 years old
```

17. Перейдите в папку `code/0hello`, используя `cd`:

```
cd ../0hello
```

18. Скомпилируйте и запустите программу `00hello.c` с помощью команды

```
gcc 00hello.c && ./a.out
```

Эта программа должна напечатать на экран `Hello World`.

Часть 2: Основы C

Hello World!

Программа на языке C, которая печатает на экран строку `Hello world` выглядит следующим образом:

```
#include <stdio.h>
int main()
{
    printf("Hello world\n");
}
```

Функции `printf` и `scanf`

- Функция `printf` используется для печати на экран всего что угодно.
- Функция `scanf` используется для считывания значений переменных с экрана.
- Обе эти функции хранятся в библиотеке `stdio.h`. Эту библиотеку нужно подключить с помощью `#include <stdio.h>`

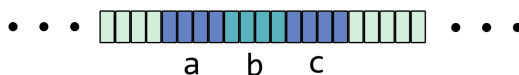
Целочисленные переменные `int`:

- Переменные типа `int` нужны для хранения целых чисел.

Адрес и размер переменной:

- 1 бит - минимальная единица измерения памяти. В 1 бите может храниться либо 0 либо 1.
- Вся память делится на ячейки, размером в 8 бит = 1 байт.
- Все эти ячейки занумерованы, номер ячейки называется адресом.
- Все переменные содержатся в памяти. Адрес переменной - это адрес первого байта переменной.
- Чтобы найти адрес переменной, нужно перед ней поставить `&`, например, `&a`
- Функции `scanf` нужно передавать именно адрес переменной, а не само значение переменной.
- Чтобы найти размер переменной в байтах: `sizeof(a)`
- Например, переменная типа `int` имеет размер 4 байта = 32 бита. Значит в ней может храниться максимум 2^{32} значений. То есть переменные типа `int` могут принимать значения от -2^{31} до 2^{31} .

`int a, b, c;`



Арифметические операторы и операторы присваивания:

+	сложение	=	присвоить левой части правую
-	вычитание	+=	прибавить к левой части правую
*	умножение	-=	отнять от левой части правую
/	целочисленное деление	/=	разделить левую часть на правую
%	остаток	%=	левая часть становится равна остатку
		++	увеличить на 1
		--	уменьшить на 1

Операторы сравнения и логические операторы:

==	равно
!=	не равно
>	больше
>=	больше или равно
<	меньше
<=	меньше или равно

&&	логическое И
	логическое ИЛИ
!	логическое НЕ

Условный оператор:

if переводится как если.

else переводится как иначе.

```
if ( условие )  
    сделай это  
else  
    сделай вот это
```

Цикл:

while переводится как пока.

```
while ( условие )  
{  
    делай это  
}
```