

# Прикладные физико-технические и компьютерные методы исследований

Семинар 2

# Что было на прошлом семинаре?

- `gcc main.c` `./a.out`
- `gcc main.c -o myprog` `./myprog`
- глобальные (инициализируются нулями по стандарту + увеличивают размер исполняемого файла)
- `i++` vs `++i`

# Что было на прошлом семинаре?

- Форматирование кода
- Именование переменных и функций  
(добавляем комментарии по необходимости)
- Указатели (Pointers)
- Stack & Heap

# Указатели

```
void Swap1(int a, int b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

```
void Swap3(int* a, int* b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
```

```
void Swap2(int* a, int* b)
{
    int* temp;
    temp = a;
    a = b;
    b = temp;
}
```

# Снова указатели ...

```
27 int main()  
28 {  
29     int x = 3;  
30     int y = 4;  
31     Swap1(x, y);  
32     // Swap2(&x, &y);  
33     // Swap3(&x, &y);  
34     printf("%d %d\n", x, y);  
35     return 0;  
36 }
```

# «Висячие» указатели (dangling)

```
// EXAMPLE 1
int *ptr = (int *)malloc(sizeof(int));
.....
.....
free(ptr);
```

```
// ptr is a dangling pointer now and operations like following are invalid
*ptr = 10; // or printf("%d", *ptr);
```

```
// EXAMPLE 2
int *ptr = NULL
{
    int x = 10;
    ptr = &x;
}
// x goes out of scope and memory allocated to x is free now.
// So ptr is a dangling pointer now.
```

# Утечка памяти

```
/* Function with memory leak */  
#include <stdlib.h>  
  
void f()  
{  
    int *ptr = (int *) malloc(sizeof(int));  
  
    /* Do some work */  
  
    return; /* Return without freeing ptr*/  
}
```

# Что выведет программа (1)?

```
# include <stdio.h>
void fun(int x)
{
    x = 30;
}

int main()
{
    int y = 20;
    fun(y);
    printf("%d", y);
    return 0;
}
```

- ☒ A 30
- ☐ B 20
- ☐ C Compiler Error
- ☐ D Runtime Error



# Что выведет программа (1)?

```
# include <stdio.h>
void fun(int x)
{
    x = 30;
}

int main()
{
    int y = 20;
    fun(y);
    printf("%d", y);
    return 0;
}
```



30



20



Compiler Error







Runtime Error

# Что выведет программа (2)?

```
# include <stdio.h>
void fun(int *ptr)
{
    *ptr = 30;
}

int main()
{
    int y = 20;
    fun(&y);
    printf("%d", y);

    return 0;
}
```

-  20
-  30
-  Compiler Error
-  Runtime Error

# Что выведет программа (2)?

```
# include <stdio.h>
void fun(int *ptr)
{
    *ptr = 30;
}

int main()
{
    int y = 20;
    fun(&y);
    printf("%d", y);

    return 0;
}
```



20



30



Compiler Error



Runtime Error

# Что выведет программа (3)?

Assume that float takes 4 bytes, predict the output of following program.

```
#include <stdio.h>
```

```
int main()  
{
```

```
    float arr[5] = {12.5, 10.0, 13.5, 90.5, 0.5};
```

```
    float *ptr1 = &arr[0];
```





```
    float *ptr2 = ptr1 + 3;
```

```
    printf("%f ", *ptr2);
```

```
    printf("%d", ptr2 - ptr1);
```

```
    return 0;
```

```
}
```

-  90.500000 3
-  90.500000 12
-  10.000000 12
-  0.500000 3

# Что выведет программа (3)?

Assume that float takes 4 bytes, predict the output of following program.

```
#include <stdio.h>
```

```
int main()
{
    float arr[5] = {12.5, 10.0, 13.5, 90.5, 0.5};
    float *ptr1 = &arr[0];
    float *ptr2 = ptr1 + 3;

    printf("%f ", *ptr2);
    printf("%d", ptr2 - ptr1);

    return 0;
}
```



90.500000 3



90.500000 12



10.000000 12



0.500000 3

# Что выведет программа (4)?

```
#include<stdio.h>
int main()
{
    int arr[] = {10, 20, 30, 40, 50, 60};
    int *ptr1 = arr;
    int *ptr2 = arr + 5;
    printf("Number of elements between two pointer are: %d.",
           (ptr2 - ptr1));
    printf("Number of bytes between two pointers are: %d",
           (char*)ptr2 - (char*) ptr1);
    return 0;
}
```

Assume that an int variable takes 4 bytes and a char variable takes 1 byte

A

Number of elements between two pointer are: 5. Number of bytes between two pointers are: 20

B

Number of elements between two pointer are: 20. Number of bytes between two pointers are: 20

C

Number of elements between two pointer are: 5. Number of bytes between two pointers are: 5

D

Compiler Error

E

Runtime Error

# Что выведет программа (4)?

```
#include<stdio.h>
int main()
{
    int arr[] = {10, 20, 30, 40, 50, 60};
    int *ptr1 = arr;
    int *ptr2 = arr + 5;
    printf("Number of elements between two pointer are: %d.",
           (ptr2 - ptr1));
    printf("Number of bytes between two pointers are: %d",
           (char*)ptr2 - (char*) ptr1);
    return 0;
}
```

Assume that an int variable takes 4 bytes and a char variable takes 1 byte



Number of elements between two pointer are: 5. Number of bytes between two pointers are: 20

B

Number of elements between two pointer are: 20. Number of bytes between two pointers are: 20

C

Number of elements between two pointer are: 5. Number of bytes between two pointers are: 5

D

Compiler Error

E

Runtime Error

# Что выведет программа (5)?

```
#include<stdio.h>
int main()
{
    int a;
    char *x;
    x = (char *) &a;
    a = 512;
    x[0] = 1;
    x[1] = 2;
    printf("%dn",a);
    return 0;
}
```

What is the output of above program?

A

Machine dependent

B

513

C

258

D

Compiler Error



# Что выведет программа (5)?

```
#include<stdio.h>
int main()
{
    int a;
    char *x;
    x = (char *) &a;
    a = 512;
    x[0] = 1;
    x[1] = 2;
    printf("%dn",a);
    return 0;
}
```

What is the output of above program?



Machine dependent



513



258



Compiler Error

Output is 513 in a little endian machine. To understand this output, let integers be stored using 16 bits. In a little endian machine, when we do  $x[0] = 1$  and  $x[1] = 2$ , the number  $a$  is changed to 00000001 00000010 which is representation of 513 in a little endian machine.

# Работа со строками в языке «С»

---

```
1 char s[] = "Hello"; s[0] = 'h'; // ok
2 char* p = "Hello"; p[0] = 'h'; // wrong
```

## String Operations <string.h>

s,t are strings, cs,ct are constant strings

- **strlen(s) length of s**
- strcpy(s,ct) copy ct to s
- strncpy(s,ct,n) up to n chars
- strcat(s,ct) concatenate ct after s
- strncat(s,ct,n) up to n chars
- strcmp(cs,ct) compare cs to ct

# Работа со строками в языке «С»

- `strncmp(cs,ct,n)` only first n chars
- `strchr(cs,c)` pointer to first c in cs
- `strrchr(cs,c)` pointer to last c in cs
- `memcpy(s,ct,n)` copy n chars from ct to s
- `memmove(s,ct,n)` copy n chars from ct to s
- (may overlap)
- `memcmp(cs,ct,n)` compare n chars of cs with ct
- `memchr(cs,c,n)` pointer to first c in first n chars of cs
- `memset(s,c,n)` put c into first n chars of cs

# Упражнение 1

Необходимо сгенерировать последовательность строк вида (выделить память, сгенерировать, а затем вывести):

- a
- a**b**a
- abac**a**ba
- abacabab**a**abacaba
- ...

```
void GenerateString(int n, char* string);
```

# Общие слова

- **Ядро ОС**

ядро координирует доступ ко всем ресурсам компьютера - память, процессорное время, файловая система, сеть.

- **Обращение к ядру через системные вызовы**

к ядру можно программно обращаться с помощью системных вызовов. Эти системные вызовы внешне ничем не отличаются от вызовов обычных функций языка C таких как `strlen()`, `strcpy()`...

0 – нет ошибки, -1 – ошибка.

NULL – ошибка.

`<errno.h>` глобальная переменная `errno` – точное значение ошибки. `perror()`.

# Общие слова

- Интерфейс системных вызовов

интерфейс у системных вызовов совпадает для все linux-подобных ос, но реализация может быть разной.

- login/password

система многопользовательская

когда набираете пароль в консоли, то может не отображаться динамика ввода

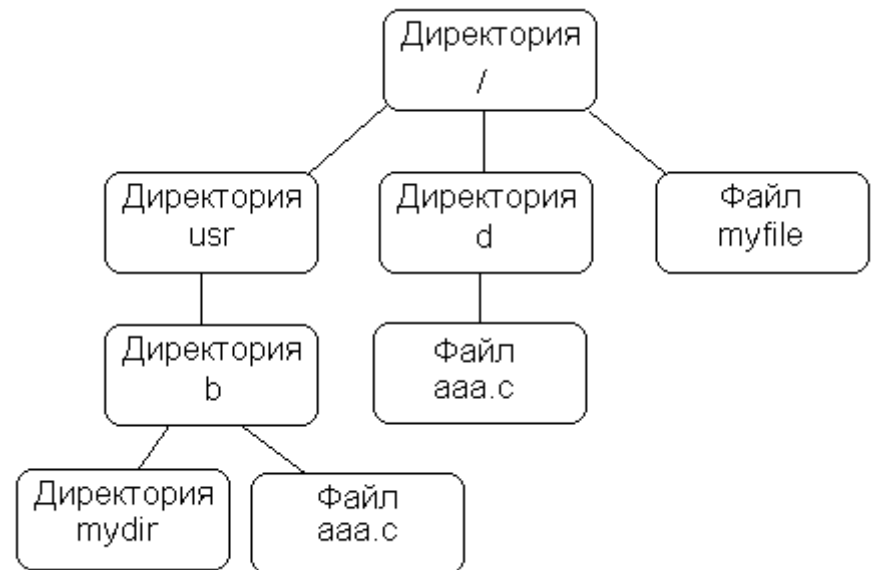
# Общие слова

- Древовидное устройство файловой системы
- Полное/относительное имя файла

полные имена файлов могут быть слишком длинными +  
если перекопировать в другое место, то все «сломается».

относительные имена файлов +  
текущая/рабочая директория (своя для  
каждого открытого окна консоли +  
для каждой программы)

- Домашняя директория (`cd ~`)
- `pwd`



# Общие слова

- справочник команд `man`

`man open / man 2 open`

`man man` 😊

- команды `cd`, `ls` (без скрытых файлов), `ls -a` (полный список)

- перенаправление ввода/вывода в файл

`./a.out > output.txt`

`./a.out < input.txt`

`./a.out < input.txt > output.txt`

`./a.out | ./b.out | ./c.out > output.txt`

**Упражнение 2:** Считать строку с консоли + вывести её.  
Потренироваться с перенаправлением ввода/вывода.



# Общие слова

- cp, rm, mkdir, mv
- регулярные выражения, вместо имён  
rm -r \*
- mc (midnight commander)
- cat файл1 файл2 ... файлN > файл
- у каждого пользователя есть свой персональный UID (User Identifier) и каждый пользователь состоит в какой-то группе GID (chown, chgrp)

# Группы пользователей и права доступа

- Пользователь, являющийся хозяином файла.
- Пользователи, относящиеся к группе хозяев файла.
- Все остальные пользователи.
- Чтение (*read*), запись (*write*), запуск (*execute*)
- *chmod +x filename, chmod 777 filename*
- *umask*
- Для директорий *+r* – читать только имена файлов в папке, *+x* – дополнительные параметры файлов + возможность сделать её текущей, *+w* – создание/удаление/переименование файлов.

## Упражнение 3

- Напишите программу, которая печатала бы идентификатор пользователя, запустившего программу, и идентификатор его группы
- `getuid()` , `getgid()`

?

- Хотим создать функцию, которая должна вернуть несколько величин ...
- Но можно только один раз написать  
**return SomeVariable;**
- *Что делать?*

?

- Создать структуру, включающую в себя все необходимые переменные.
- Передать в функцию указатели на переменные, по которым запишутся «выходные» результаты.

?

```
void f(int x, int* x2, int* x3)
{
    *x2 = x * x;
    *x3 = x * x * x;
}
```

```
int main()
{
    int x2;
    int x3;

    f(3, &x2, &x3);
}
```

# Как правильно объявить функцию?

- Нужна функция `Split`, принимающая строку (1) и разделительные символы (2); возвращающая массив «слов» в исходной строке (3).
- Например,
  1. «Мама мыла раму»
  2. « \t »
  3. «Мама», «мыла», «раму»

# Как правильно объявить функцию?

```
void Split(char* string,  
           char* delimiters,  
           char*** tokens,  
           int* tokensCount);
```



# Как реализовать функцию Split?

- `man strtok`

# Упражнение 4

- Реализовать разбиение строки на «слова»
  - На вход строка (scanf/gets).
  - Вывести в столбик «слова» строки.
  - Вынести код разбиения в отдельную функцию + продемонстрировать её работу.

Если не успели на семинаре, то нужно сделать упражнения до следующего занятия: крайний срок сдачи – следующий семинар.

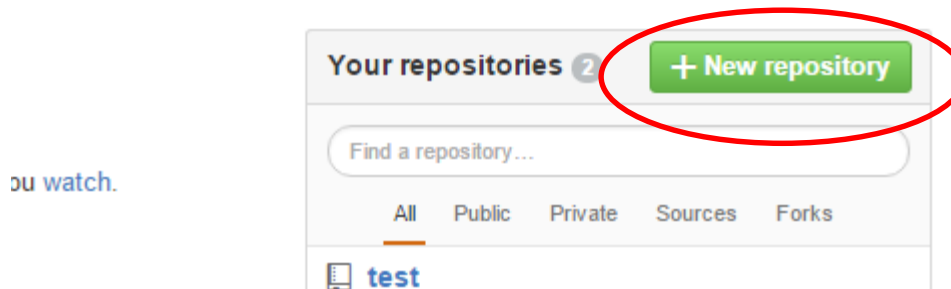
Также необходимо ознакомиться и потренироваться работать с Git'ом.

# Сдача упражнений


- Система контроля версий git
- Репозиторий на [github.com](https://github.com)

# Git

- Регистрируемся на github.com
- Создаём репозиторий для всех своих домашних упражнений




# Git


Owner:  VladMiryaha ▾

Repository name:  ✓

Great repository names are short and memorable. Need inspiration? How about...

Description (optional):

☒  **Public**  
Anyone can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip this step if you already have a README file.

Add .gitignore: **None** ▾ | Add a license: **None** ▾ ⓘ

**Create repository**

# Git

- Public означает, что любой сможет посмотреть и выкачать ваш репозиторий
- Чтобы кто-то ещё смог вносить изменения, надо добавить пользователя в список *collaborator`*ов.

<https://help.github.com/articles/adding-outside-collaborators-to-repositories-in-your-organization/>

# Git

- Установить git (для Ubuntu: `sudo apt-get install git`)

- Выкачивание репозитория на свою машину:

```
git clone git://github.com/VladMiryaha/test.git
```

- Добавляем файл в репозиторий

```
git add hw01.c
```

```
git commit -m "deadline 07.09.15"
```

```
git push
```

- Чтобы выкачать из репозитория изменения

```
git pull
```

- Последние изменения

```
git diff HEAD@{1}
```

```
(git log -p 2)
```

- Состояние репозитория

```
git status
```

# Git

Для понимания работы git'а нужно прочитать первые 2 главы книги

<https://git-scm.com/book/ru/v1/>

и пройти

<https://try.github.io>



# Git

## File Status Lifecycle

