Лекция 14. Хранение, связывание и управление памятью.

Ревун Артем Леонидович
ст. преп. кафедры вычислительных систем

Курс «Программирование», весенний семестр, 2024
Сибирский государственный университет телекоммуникаций и информатики (г. Новосибирск)

Содержание

- Ключевые слова: ***auto, extern, static, register, const, volatile, restricted, _Thread_local, _Atomic***
- Функции: ***rand(), srand( ), time( ), malloc( ), calloc(), free()***
- Определение в языке С области видимости переменной и времени жизни переменной
- Проектирование более сложных программ

Modern C (Jens Gustedt) - C23

HAL open science

Modern C

Jens Gustedt

https://inria.hal.science/hal-02383654v2/file/modernC.pdf

Ключевые понятия (1)

Объект – …

Идентификатор – …

L-значение – …

Модифицируемое L-значение – …

Область видимости – …

Связывание – …

Продолжительность хранения – …

Ключевые понятия (2)

Объект – описание некоторого участка физической памяти, *определенного размера*, который может хранить одно или более значений. В зависимости от момента, объект может **содержать** или **не содержать** *сохраненного* значения.

```
int initVal = 0, val[5];
 //4byte          20byte
```

Ключевые понятия (3)

Идентификатор – – последовательность строчных (**"a – z"**) и прописных (**"A – Z"**) букв латинского алфавита, а также цифр (**"0 – 9"**) и знака подчеркивания ("_").

Идентификатор ***начинается*** либо ***с буквы***, либо ***со знака нижнего*** подчеркивания.

Но это не всё!

Ключевые понятия (4)

Идентификатор – отображает способ, которым программа написанная на Си, указывает на *объект*, хранящийся в аппаратной памяти. И регламентирует действия, которые могут быть выполнены с *объектом*.

```
int vals[10];  const int valConst[10];
int *pt=&vals; const *int ptConst=&vals;
```

Ключевые понятия (5)

L-значение – выражение которое обозначает *объект*. Чаще всего *L-значение* представлено идентификатором или выражением.

```
int val = 10;
int *pt=&val;
```

Ключевые понятия (6)

Модифицируемое L-значение – L-значение, которое может быть использовано для изменения *значения* внутри объекта.

```
const char * text = "Text";
```

Ключевые понятия (7)

Область видимости (ОВ) – описывает участок или участки программ, где можно обращаться к идентификатору. Например:

- в пределах блока
- в пределах функции
- в пределах прототипа функции
- в пределах файла

Ключевые понятия:область видимости (8)

Область видимости: в пределах блока

```
{
double val = 10;
...
return val;
}
```

Ключевые понятия:область видимости (9)

Область видимости: в пределах блока

```
double blocky(double cval){
  double val = 0.; int i;
  for (i = 0; i < 10; i++){
    double q = cval * i; val *= q;
  }
  return val;
}
```

Ключевые понятия:область видимости (10)

Область видимости: в пределах функции применяется только к меткам операции **goto.**

```
void someFunct(int a, int b){
    if (a > b) goto end;
    printf("%d\n", b);
    end: printf("%d\n", a);
}
```

Ключевые понятия:область видимости (11)

Область видимости: в пределах прототипа функции применяется к переменным используемым в прототипах.

```
void someFunct(int a, int b);


void someArrayFucnt(int n, int m,
                    array[n][m]);
```

Ключевые понятия:область видимости (12)

Область видимости: в пределах файла

```
int units = 0;
int main(void){
    /* code */
    return 0;
}
```

Глобальные переменные

Ключевые понятия: связывание (13)

В Си переменная имеет одно из следующих связываний:

- внешнее связывание
- внутреннее связывание
- отсутствие связывания

Связывание – описывает видимость идентификаторов программы.

Ключевые понятия: связывание (14)

Отсутствие связывания:

Переменные с **ОВ** в пределах блока, функции или прототипа функции. Они являются закрытыми для блока, функции или прототипа, в котором определены.

```
{    // void someFunct(int closed);
     int closed = 1;
     ...
}
```

Ключевые понятия: связывание (15)

Внутреннее связывание:

Переменные с **ОВ** в пределах файла.

Для внутреннего связывания применяется ключевое слово static, которое явно указывает свяойство связывание и запрещает к использованию переменно в другом файле.

```
static char ch = '/n';
static void printSome(void){
    puts("Some");
}
```

Ключевые понятия: связывание (16)

Внешнее связывание:

Переменные с **ОВ** в пределах файла.

```
int units = 0;
int main(void){
    /* code */
    return 0;
}
```

Ключевые понятия (17)

Продолжительность хранения (ПХ) – регламентирует постоянство объектов, доступных через эти идентификаторы.

Выделяют:

- статическое
- потоковое
- автоматическое
- выделенное

Ключевые понятия: продолжительность хранения (18)

Статическая ПХ: объект существует на протяжении времени выполнения программы.

Например: переменная с ОВ в пределах файла.

```
int units = 0;
int main(void){
    /* code */
    return 0;
}
```

Ключевые понятия: продолжительность хранения (19)

Потоковая ПХ: объект существует ровно столько, с момента его объявления и до завершения потока.
Поток – часть программы, которая выполняется внутри процесса.

```
void *threadFunc(void) {
    int arg = 42;
    return NULL;}
for (size_t i = 0; i < 4; i++)
    pthread_create(&thread, NULL,
                   thread_function, &arg);
```

Ключевые понятия: продолжительность хранения (20)

Потоковая ПХ: объект существует ровно столько, с момента его объявления и до завершения потока.

```
//thread #1 start    int arg = 42; end
//thread #2 start    int arg = 42; end
//thread #3 start    int arg = 42; end
//thread #4 start    int arg = 42; end
```

Ключевые понятия: продолжительность хранения (21)

Автоматическая ПХ: объект существует в пределах блока или в пределах функции.

```
for (size_t i = 0; i < 4; i++){
    int otherArg = 0;
}
void someFunct(void){
    int arg = 0;
}
```

Ключевые понятия: продолжительность хранения (22)

Если возникает потребность изменить ***автоматическую*** ПХ на ***статическую*** следует использовать ключевое слово **static**

```
void someFunct(void){
    int arg = 0;
    static int otherArg = 0;
}
```

Ключевые понятия: продолжительность хранения (23)

Если возникает потребность изменить ***автоматическую*** ПХ на ***статическую*** следует использовать ключевое слово **static**

```
void someFunct(void){
    int arg = 0;
    static int otherArg = 0;
}
```

Пять классов хранения

Класс хранения	Продолжительность хранения	Область видимости	Связывание	Объявление
Автоматический	Автоматическая	В пределах блока	Нет	В блоке
Регистровый	Автоматическая	В пределах блока	Нет	В блоке с указанием ключевого слова `register`
Статический с внешним связыванием	Статическая	В пределах файла	Внешнее	За рамками всех функций
Статический с внутренним связыванием	Статическая	В пределах файла	Внутреннее	За рамками всех функций с указанием ключевого слова `static`
Статический без связывания	Статическая	В пределах файла	Нет	В блоке с указанием ключевого слова `static`

Автоматические переменные (1)

Любая переменная, объявленная в блоке или заголовке функции, относится к автоматическому классу хранения. Однако, можно явно указать класс хранения используя ключевое слово auto.

```
int main() {
    auto int m;
}
```

auto - спецификатор класса хранения.

Автоматические переменные (2)

Автоматические переменные можно инициализировать только явным образом.

```
int main(void)
{
    int repid;
    int tents = 5;
}
```

Автоматические переменные (3)

```
int main(void){
    int x = 30; // исходная переменная x
    {
      int x = 77; // новая переменная x,
                  скрывающая первую x
    }
}
```

Регистровые переменные (1)

Си даёт возможность разместить значение переменной в регистровой памяти процессора, при благоприятных условиях. У такой переменной нельзя получить адрес. Для объявления используется спецификатор класса хранения **register**.

```
int main(void){
    register int quick;
}
```

Статическая переменная с областью видимости в пределах блока (1)

Такая переменная имеет область видимости в пределах блока, но компьютер запоминает её значение от одного вызова функции до следующего. Другими словами она имеет статическую (**static**) продолжительность хранения.

Однако, использование модификатора **static** запрещено для параметров функции

```
int funct(static int arg);
// не разрешено
```

Статическая переменная с областью видимости в пределах блока (2)

```
void trystat(void){
    int fade = 1;
    static int stay = 1;
    printf ("fade = %d и stay = %d\n",
                    fade ++, stay++);
}
trystat(); //fade = 1 и stay = 1
trystat(); //fade = 1 и stay = 2
trystat(); //fade = 1 и stay = 3
```

Статическая переменная с внешним связыванием (1)

Такая переменная имеет область видимости в пределах файла, внешнюю связанность и статическую продолжительность хранения. Для объявления таких переменных используется ключевое слово **extern**. Такие переменные по другому называют ***внешними,*** по типу связанности.

Статическая переменная с внешним связыванием (2)

```
int Errupt;//внеш. опред. переменная
double Up[100];//внеш. опред. массив
extern char Coal;//если опред. в другом
файле */
int main(void){
    extern int Errupt;
    extern double Up[];
}
```

Статическая переменная с внешним связыванием (3)

```
int Hocus;
int magic();
int main (void){
    extern int Hocus;
};
int magic (){
    extern int Hocus;
}
```

Статическая переменная с внешним связыванием (4)

```
int Hocus;
int magic();
int main (void){
    extern int Hocus;
};
int magic (){
    // Hocus;
}
```

Статическая переменная с внешним связыванием (5)

```
int Hocus;
int magic();
int main (void){
    int Hocus;  }
int Pocus;
int magic (){
    auto int Hocus;    }
```

Инициализация внешних переменных(1)

Могут быть инициализированы только явно. Если они не были инициализированы, они инициализируются нулём, в том числе и массивы. Однако, для инициализации могут быть использованы только константные выражения.

```
int x = 10;
int y=3+20;
size_t z = sizeof(int);
int x2 = 2 * x;
```

Инициализация внешних переменных(2)

Внешняя переменная должна быть инициализирована, только один раз.

```
// файл one.c
char permis = 'N';
// файл two. c
extern char permis = 'Y'; /* ошибка */
```

По сути своей extern указывает, что вы ссылаетесь на уже существующую переменную, а не создаёте новую.

Статические переменные с внутренним связыванием(1)

Переменные имеют статическую продолжительность хранения, область видимости в приделах файла и внутреннее связывание. Создание такой переменной предполагает использование ключевого слова static, размещение вне функции и блока.

```
static int val = 1;
int main(void)

…
```

Статические переменные с внутренним связыванием(2)

```
int traveler = 1;
static int stayhome = 1;
int main ()
{
    extern int traveler;
    extern int stayhome;
}
```

Продолжение следует…
Спасибо за внимание

Ревун Артем Леонидович
ст. преп. кафедры вычислительных систем

Курс «Программирование», весенний семестр, 2024
Сибирский государственный университет телекоммуникаций и информатики (г. Новосибирск)