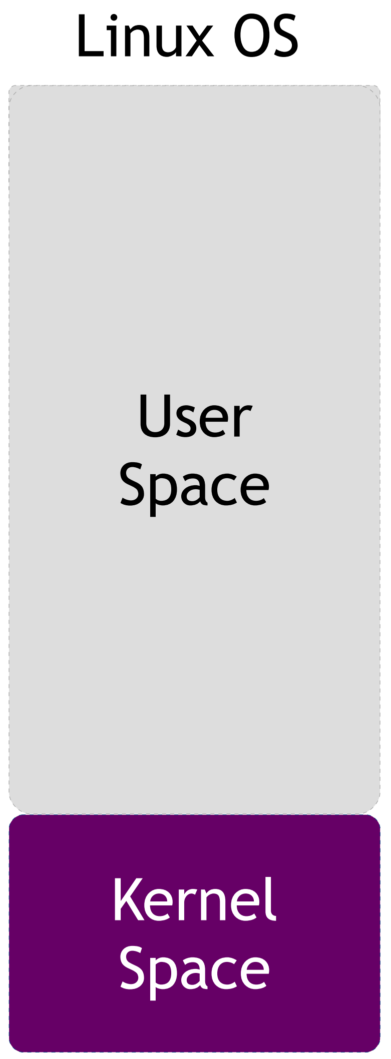
C
Come let's see how deep it is!!
- Storage Classes

Team Emertxe

Advanced C

Memory Segments



The Linux OS is divided into two major sections

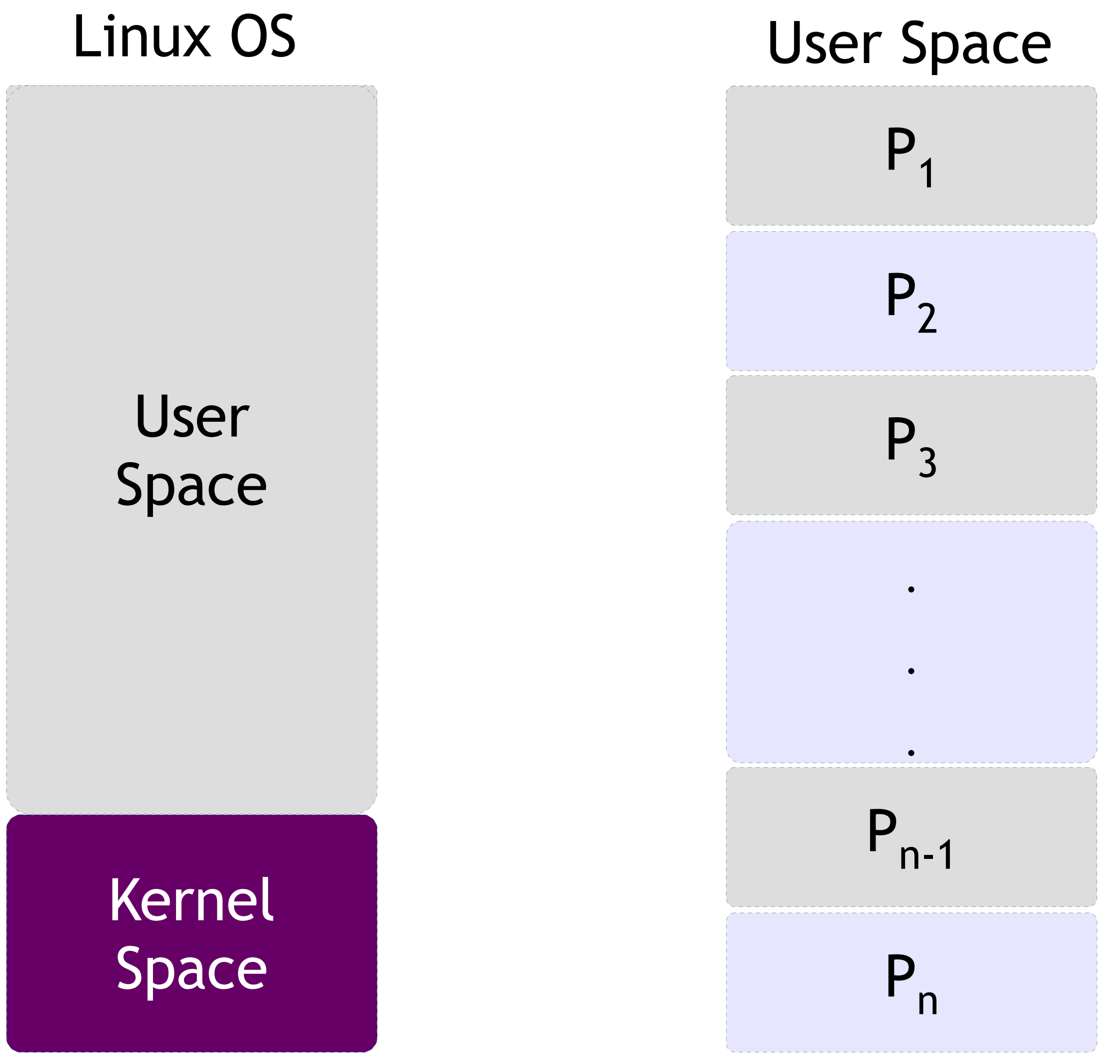
- User Space
- Kernel Space

The user programs cannot access the kernel space. If done will lead to segmentation violation

Let us concentrate on the user space section here

Advanced C

Memory Segments



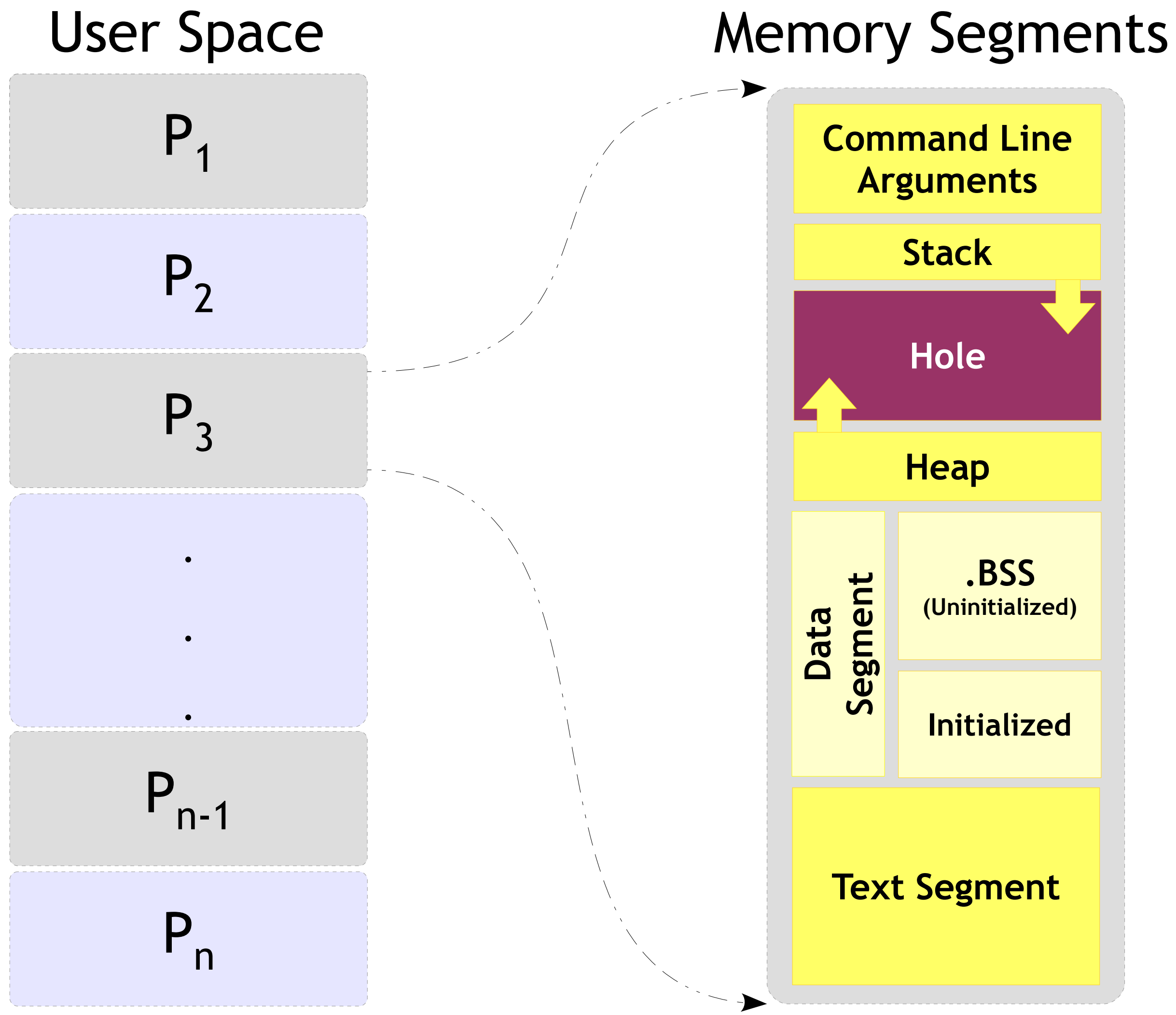
The User space contains many processes

Every process will be scheduled by the kernel

Each process will have its memory layout discussed in next slide

Advanced C

Memory Segments



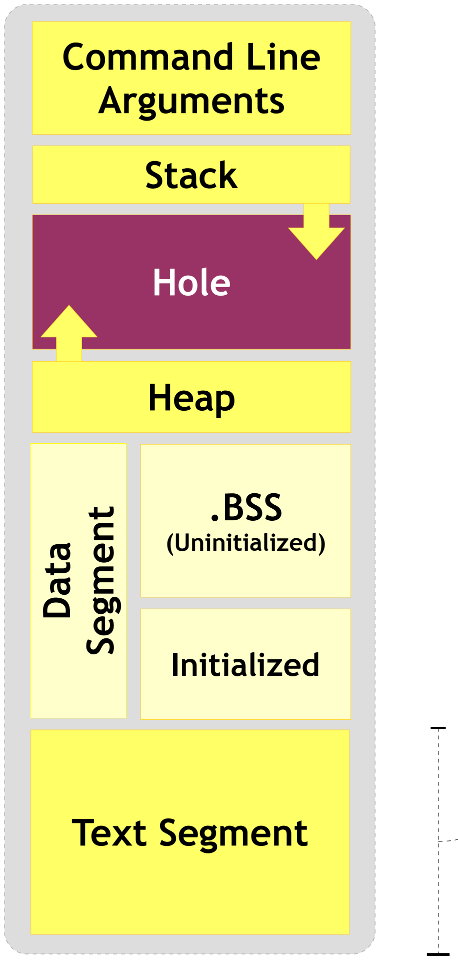
The memory segment of a program contains four major areas.

- Text Segment
- Stack
- Data Segment
- Heap

Advanced C

Memory Segments – Text Segment

Memory Segments

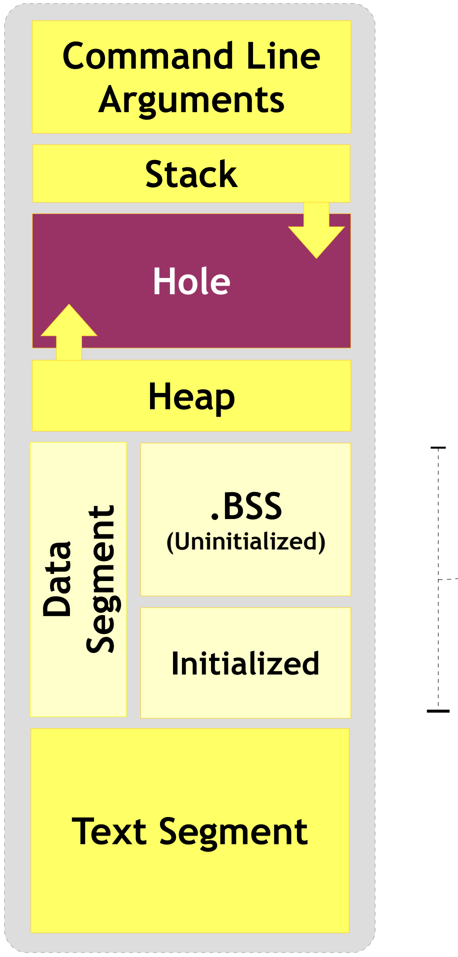


- Also referred as Code Segment
- Holds one of the section of program in object file or memory
- In memory, this is place below the heap or stack to prevent getting over written
- Is a read only section and size is fixed

Advanced C

Memory Segments – Data Segment

Memory Segments

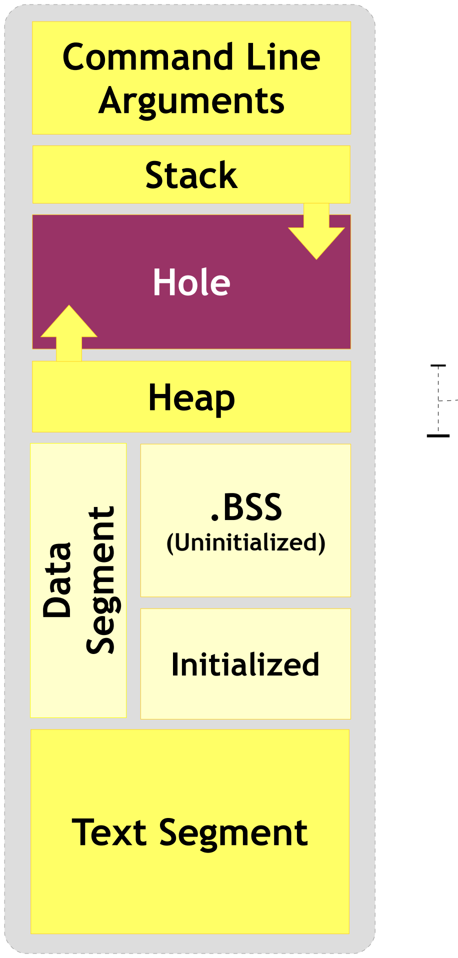


- Contains 2 sections as initialized and uninitialized data segments
- Initialized section is generally called as Data Segment
- Uninitialized section is referred as BSS (Block Started by Symbol) usually filled with 0s

Advanced C

Memory Segments – Data Segment

Memory Segments



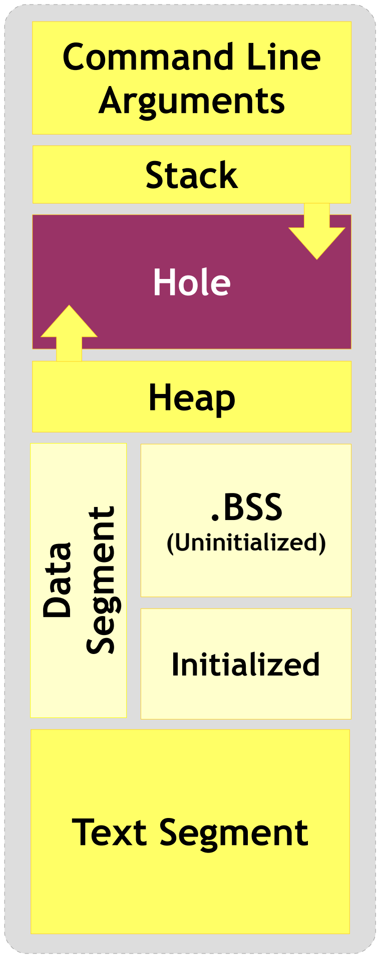
Dynamic memory allocation takes place here

Begins at the end of BSS and grows upward from there

Advanced C

Memory Segments – Stack Segment

Memory Segments



Adjoins the heap area and grow in opposite area of heap when stack and heap pointer meet (Memory Exhausted)

Typically loaded at the higher part of memory

A “stack pointer” register tracks the top of the stack; it is adjusted each time a value is “pushed” onto the stack

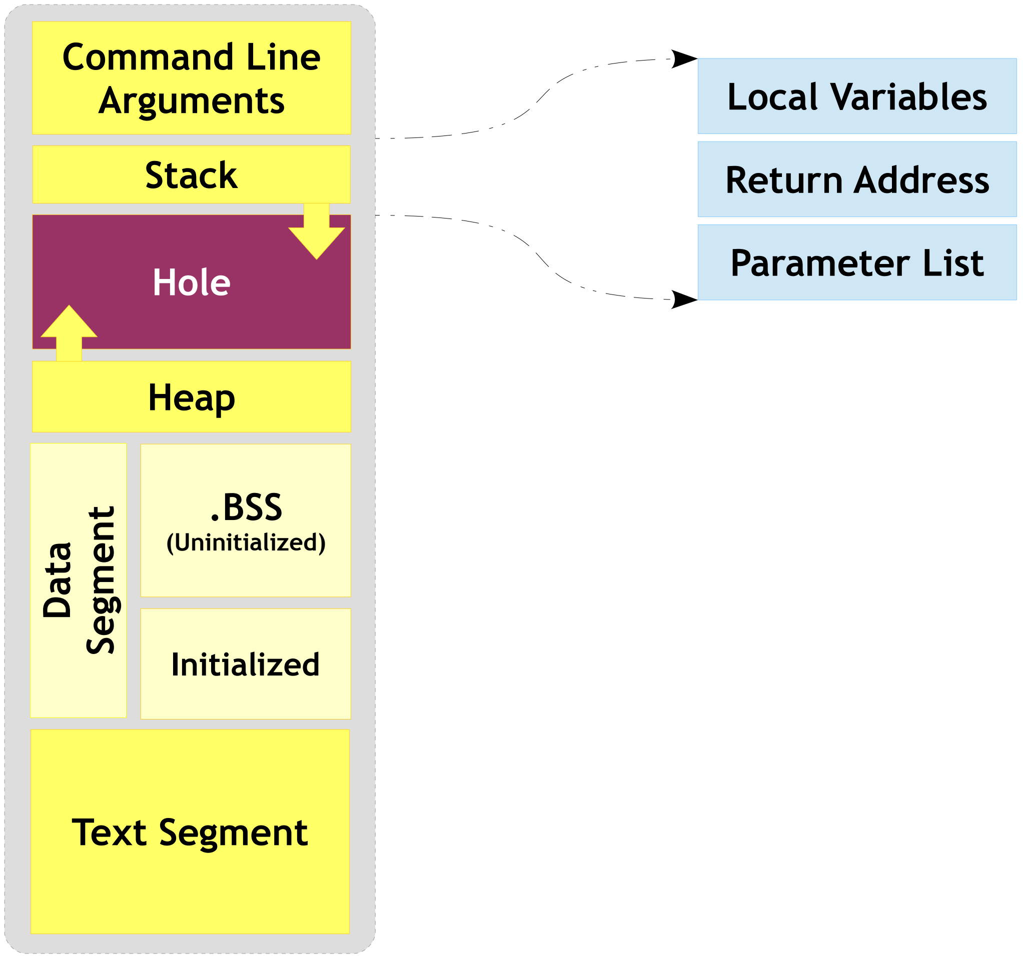
The set of values pushed for one function call is termed a “stack frame”

Advanced C

Memory Segments – Stack Segment

Memory Segments

Stack Frame



A stack frame contain at least of a return address

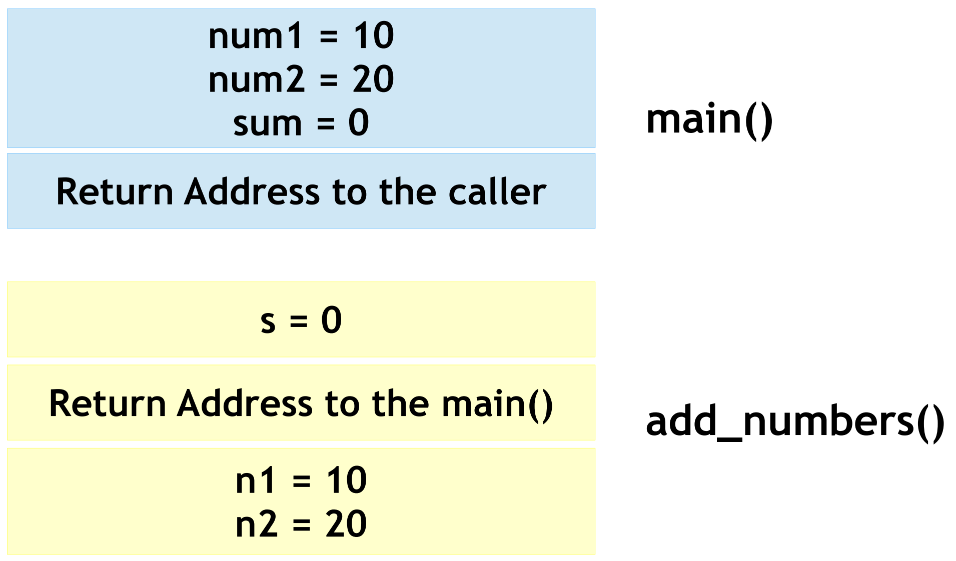
Advanced C

Memory Segments - Stack Frame

```c
#include <stdio.h>

int main()
{
    int num1 = 10, num2 = 20;
    int sum = 0;

    sum = add_numbers(num1, num2);
    printf(“Sum is %d\n”, sum);

    return 0;
}
```

```c
int add_numbers(int n1, int n2)
{
    int s = 0;

    s = n1 + n2;

    return s;
}
```

Stack Frame



Advanced C

Memory Segments - Runtime

- Text Segment: The text segment contains the actual code to be executed. It's usually sharable, so multiple instances of a program can share the text segment to lower memory requirements. This segment is usually marked read-only so a program can't modify its own instructions
- Initialized Data Segment: This segment contains global variables which are initialized by the programmer
- Uninitialized Data Segment: Also named “BSS" (block started by symbol) which was an operator used by an old assembler. This segment contains uninitialized global variables. All variables in this segment are initialized to 0 or NULL (for pointers) before the program begins to execute

Advanced C

Memory Segments - Runtime

- The Stack: The stack is a collection of stack frames. When a new frame needs to be added (as a result of a newly called function), the stack grows downward
- The Heap: Most dynamic memory, whether requested via C's malloc() . The C library also gets dynamic memory for its own personal workspace from the heap as well. As more memory is requested "on the fly", the heap grows upward

Advanced C

Storage Classes

Variable	Storage Class	Scope	Lifetime	Memory Allocation	Linkage
Local	auto	Block	B/W block entry and exit	Stack	None
	register	Block	B/W block entry and exit	Register / Stack	None
	static	Block	B/W program start & end	Data Segment	None
Global	static	File	B/W program start & end	Data segment	Internal
	extern	Program	B/W program start & end	Data segment	Internal / External
Function Parameter	register	Function	Function entry and exit	Stack	None

***Block : function body or smaller block with in function**

Advanced C

Declaration

```c
extern int num1;
extern int num1;

int main();

int main()
{
    int num1, num2;
    char short_opt;

    ...
}
```

Declaration specifies type to the variables

Its like an announcement and hence can be made 1 or more times

Declaration about num1

Declaration about num1 yet again!!

Declaration about main function

Advanced C

Declaration

```
extern int num1;
extern int num1;

int main();

int main()
{
    int num1, num2;
    char short_opt;

    ...
}
```

Declaration specifies type to the variables

Its like an announcement and hence can be made 1 or more times

Declaration about num1

Declaration about num1 yet again!!

Declaration about main function

*** One Definition Rule for variables ***
"In a given scope, there can be only one definition of a variable"

Advanced C

Storage Classes - Auto

001_example.c

```
#include <stdio.h>

int main()
{
    int i = 0;

    printf(“i %d\n”, i);

    return 0;
}
```

Advanced C

Storage Classes - Auto

002_example.c

```
#include <stdio.h>

int foo()
{
    int i = 0;

    printf("i %d\n", i);

    return 0;
}

int main()
{
    foo();

    return 0;
}
```

Advanced C

Storage Classes - Auto

003_example.c

```c
#include <stdio.h>

int *foo()
{
    int i = 10;
    int *j = &i;

    return j;
}

int main()
{
    int *i;

    i = foo();
    printf("*i %d\n", *i);

    return 0;
}
```

Advanced C

Storage Classes - Auto

004_example.c

```
#include <stdio.h>

char *foo()
{
    char ca[12] = “Hello World”;

    return ca;
}

int main()
{
    char *ca;

    ca = foo();
    printf(“ca is %s\n”, ca);

    return 0;
}
```

Advanced C

Storage Classes - Auto

005_example.c

```c
#include <stdio.h>

int book_ticket()
{
    int ticket_sold = 0;

    ticket_sold++;

    return ticket_sold;
}

int main()
{
    int count;

    count = book_ticket();
    count = book_ticket();

    printf("Sold %d\n", count);

    return 0;
}
```

Advanced C

Storage Classes - Auto

006_example.c

```c
#include <stdio.h>

int main()
{
    int i = 0;

    {
        int j = 0;

        printf(“i %d\n”, i);
    }

    printf(“j %d\n”, j);

    return 0;
}
```

Advanced C

Storage Classes - Auto

007_example.c

```
#include <stdio.h>

int main()
{
    int j = 10;

    {
        int j = 0;

        printf(“j %d\n”, j);
    }

    printf(“j %d\n”, j);

    return 0;
}
```

Advanced C

Storage Classes - Auto

008_example.c

```
#include <stdio.h>

int main()
{
    int i = 10;
    int i = 20;

    {
        printf("i %d\n", i);
    }

    printf("i %d\n", i);

    return 0;
}
```

Advanced C

Storage Classes - Register

009_example.c

```
#include <stdio.h>

int main()
{
    register int i = 0;

    scanf("%d", &i);
    printf("i %d\n", i);

    return 0;
}
```

Advanced C

Storage Classes - Register

010_example.c

```
#include <stdio.h>

int main()
{
    register int i = 10;
    register int *j = &i;

    printf("*j %d\n", *j);

    return 0;
}
```

Advanced C

Storage Classes - Register

011_example.c

```
#include <stdio.h>

int main()
{
    int i = 10;
    register int *j = &i;

    printf("*j %d\n", *j);

    return 0;
}
```

Advanced C

Storage Classes – Static Local

012_example.c

```c
#include <stdio.h>

int *foo()
{
    static int i = 10;
    int *j = &i;

    return j;
}

int main()
{
    int *i;

    i = foo();
    printf(“*i %d\n”, *i);

    return 0;
}
```

Advanced C

Storage Classes - Static Local

013_example.c

```
#include <stdio.h>

char *foo()
{
    static char ca[12] = “Hello World”;

    return ca;
}

int main()
{
    char *ca;

    ca = foo();
    printf(“ca is %s\n”, ca);

    return 0;
}
```

Advanced C

Storage Classes – Static Local

014_example.c

```c
#include <stdio.h>

int book_ticket()
{
    static int ticket_sold = 0;

    ticket_sold++;

    return ticket_sold;
}

int main()
{
    int count;

    count = book_ticket();
    count = book_ticket();

    printf("Sold %d\n", count);

    return 0;
}
```

Advanced C

Storage Classes - Static Local

015_example.c

```
#include <stdio.h>

int main()
{
    static int i = 5;

    if (--i)
    {
        main();
    }

    printf("i %d\n", i);

    return 0;
}
```

016_example.c

```
#include <stdio.h>

int main()
{
    static int i = 5;

    if (--i)
    {
        return main();
    }

    printf("i %d\n", i);

    return 0;
}
```

Advanced C

Storage Classes - Static Local

017_example.c

```c
#include <stdio.h>

int foo()
{
    static int i;

    return i;
}

int main()
{
    static int x = foo();

    printf(“x %d\n”, x);

    return 0;
}
```

Advanced C

Storage Classes – Static Local

018_example.c

```c
#include <stdio.h>

int *foo()
{
    static int i = 10;
    int *j = &i;

    return j;
}

int main()
{
    int *i;

    i = foo();
    printf("*i %d\n", *i);

    return 0;
}
```

Advanced C

Storage Classes - Static Local

019_example.c

```c
#include <stdio.h>

int *foo()
{
    int i = 10;
    static int *j = &i;

    return j;
}

int main()
{
    int *i;

    i = foo();
    printf("*i %d\n", *i);

    return 0;
}
```

Advanced C

Storage Classes – Global

020_example.c

```
#include <stdio.h>

int x;

int foo()
{
    printf("x %d\n", x);

    return ++x;
}

int main()
{
    foo();

    printf("x %d\n", x);

    return 0;
}
```

Advanced C

Storage Classes - Global

021_example.c

```
#include <stdio.h>

auto int x;

int foo()
{
    printf("x %d\n", x);

    return ++x;
}

int main()
{
    foo();

    printf("x %d\n", x);

    return 0;
}
```

Advanced C

Storage Classes - Global

022_example.c

```
#include <stdio.h>

register int x;

int foo()
{
    printf("x %d\n", x);

    return ++x;
}

int main()
{
    foo();

    printf("x %d\n", x);

    return 0;
}
```

Advanced C

Storage Classes - Global

023_example.c

```
#include <stdio.h>

int x = 10;

int foo()
{
    printf("x %d\n", x);

    return 0;
}

int main()
{
    foo();

    return 0;
}
```

Advanced C

Storage Classes - Global

024_example.c

```c
#include <stdio.h>

int x = 10;
int x;

int foo()
{
    printf("x %d\n", x);

    return 0;
}

int main()
{
    foo();

    return 0;
}
```

- Will there be any compilation error?

Advanced C

Storage Classes – Global

025_example.c

```
#include <stdio.h>

int x = 10;
int x = 20;

int foo()
{
    printf(“x %d\n”, x);

    return 0;
}

int main()
{
    foo();

    return 0;
}
```

- Will there be any compilation error?

Advanced C

Storage Classes - Global

Example

```
#include <stdio.h>

int x = 10;          // Definition
int x;               // Tentative definition
extern int x;        // not a definition either
extern int x = 20;   // Definition
static int x;        // Tentative definition
```

- Declaration Vs definition
  - All the above are declarations
  - Definitions as mentioned in the comment

Advanced C

Storage Classes – Global

- Complying with one definition rule
  - All the tentative definitions and extern declarations are eliminated and mapped to definition by linking method
  - If there exists only tentative definitions then all of them are eliminated except one tentative definition which is changed to definition by assigning zero to the variable

- Compilation error if there are more then one definitions
- Compilation error if no definition or tentative definition exists

Advanced C

Storage Classes – Global

- **Translation unit -** A source file + header file(s) forms a translation unit
- Compiler translates source code file written in ‘C’ language to machine language
- A program is constituted by the set of translation units and libraries
- An identifier may be declared in a scope but used in different scopes (within a translation unit or in other translation units or libraries)
- **Linkage -** An identifier declared in different scopes or in the same scope more than once can be made to refer to the same object or function by a process called linkage
- There are three type of linkages – **internal, external and none**

Advanced C

Storage Classes – Global

- **External Linkage** – A global variable declared without storage class has "external" linkage
- **Internal Linkage** – A global variable declared with static storage class has "internal" linkage
- **None Linkage** – Local variables have "none" linkage
- Variable declaration with "extern" – in such case, linkage to be determined by referring to **"previous visible declaration"**
  - If there is no "previous visible declaration" then external linkage
  - If "previous visible declaration" is local variable then external linkage
  - If "previous visible declaration" is global variable then linkage is same as of global variable

"Compilation error if there is linkage disagreement among definition and tentative definitions"

Advanced C

Storage Classes – Static Global

026_example.c

```c
#include <stdio.h>

static int x = 10;

int foo()
{
    printf("x %d\n", x);

    return 0;
}

int main()
{
    foo();

    return 0;
}
```

Advanced C

Storage Classes - Static Global

027_example.c

```c
#include <stdio.h>

static int x = 10;
int x;

int foo()
{
    printf("x %d\n", x);

    return 0;
}

int main()
{
    foo();

    return 0;
}
```

Advanced C

Storage Classes - External

028_file1.c

```c
#include <stdio.h>

int num;

int main()
{
    while (1)
    {
        num++;
        func_1();
        sleep(1);
        func_2();
        sleep(1);
    }

    return 0;
}
```

029_file2.c

```c
#include <stdio.h>

extern int num;

int func_1()
{
    printf("num is %d from file2\n", num);

    return 0;
}
```

030_file3.c

```c
#include <stdio.h>

extern int num;

int func_2()
{
    printf("num is %d from file3\n", num);

    return 0;
}
```

ΣMERTXE

Advanced C

Storage Classes – External

032_file2.c

```
#include <stdio.h>

extern int num;
extern int num;

int func_1()
{
    printf("num is %d from file2\n", num);

    return 0;
}
```

031_file1.c

```
#include <stdio.h>

int num;

int main()
{
    while (1)
    {
        num++;
        func_1();
        sleep(1);
    }

    return 0;
}
```

Advanced C

Storage Classes - External

034_file2.c

```
#include <stdio.h>

static int num;
extern int num;

int func_1()
{
    printf("num is %d from file2\n", num);

    return 0;
}
```

033_file1.c

```
#include <stdio.h>

int num;

int main()
{
    while (1)
    {
        num++;
        func_1();
        sleep(1);
    }

    return 0;
}
```

Advanced C

Storage Classes – External

036_file2.c

```
#include <stdio.h>

extern char num;

int func_1()
{
    printf(“num is %d from file2\n”, num);

    return 0;
}
```

035_file1.c

```
#include <stdio.h>

int num;

int main()
{
    while (1)
    {
        num++;
        func_1();
        sleep(1);
    }

    return 0;
}
```

Advanced C

Storage Classes – External

038_file2.c

```c
#include <stdio.h>

extern int num;
extern char num;

int func_1()
{
    printf("num is %d from file2\n", num);

    return 0;
}
```

Conflicting types

037_file1.c

```c
#include <stdio.h>

int num;

int main()
{
    while (1)
    {
        num++;
        func_1();
        sleep(1);
    }

    return 0;
}
```

Advanced C

Storage Classes - External

039_example.c

```c
#include <stdio.h>

int main()
{
    int x;

    {
        int x = 10;
        {
            extern int x;
            printf("x %d\n", x);
        }
        printf("x %d\n", x);
    }
    printf("x %d\n", x);

    return 0;
}

int x = 20;
```

Advanced C

Storage Classes - External

040_example.c

```
#include <stdio.h>

int main()
{
    extern char x;

    printf("x %c\n", x);

    return 0;
}

int x = 0x31;
```

Advanced C

Storage Classes - External

041_example.c

```
#include <stdio.h>

int main()
{
    int x;

    {
        int x = 10;
        {
            extern int x = 20;
            printf("x %d\n", x);
        }
        printf("x %d\n", x);
    }
    printf("x %d\n", x);

    return 0;
}

int x;
```

Invalid, extern and initializer not permitted in block scope

Advanced C

Storage Classes – Static Function

042_file1.c

```c
#include <stdio.h>

int num;

int main()
{
    while (1)
    {
        num++;
        func_1();
    }

    return 0;
}
```

043_file2.c

```c
#include <stdio.h>

extern int num;

static int func_2()
{
    printf(“num is %d from file2\n”, num);

    return 0;
}

int func_1()
{
    func_2();
}
```

Advanced C

Storage Classes – Static Function

045_file2.c

```c
#include <stdio.h>

extern int num;

static int func_2()
{
    printf("num is %d from file2\n", num);

    return 0;
}

int func_1()
{
    func_2();
}
```

044_file1.c

```c
#include <stdio.h>

int num;

int main()
{
    while (1)
    {
        num++;
        func_2();
    }

    return 0;
}
```

Extra Examples

Advanced C

Storage Classes – External

file2.c

```
#include <stdio.h>

extern int num;
static int num;

int func_1()
{
    printf(“num is %d from file2\n”, num);

    return 0;
}
```

file1.c

```
#include <stdio.h>

int num;

int main()
{
    while (1)
    {
        num++;
        func_1();
        sleep(1);
    }

    return 0;
}
```

- Compiler error due to linkage disagreement

Advanced C

Storage Classes – External

file2.c

```
#include <stdio.h>

int num;
static int num;

int func_1()
{
    printf(“num is %d from file2\n”, num);

    return 0;
}
```

file1.c

```
#include <stdio.h>

int num;

int main()
{
    while (1)
    {
        num++;
        func_1();
        sleep(1);
    }

    return 0;
}
```

- Compiler error due to linkage disagreement

Advanced C

Storage Classes – External

file2.c

```
#include <stdio.h>

static int num;
int num;

int func_1()
{
    printf("num is %d from file2\n", num);

    return 0;
}
```

file1.c

```
#include <stdio.h>

int num;

int main()
{
    while (1)
    {
        num++;
        func_1();
        sleep(1);
    }

    return 0;
}
```

- Compiler error due to linkage disagreement

Advanced C

Storage Classes - External

Example

```
#include <stdio.h>

int main()
{
    int x;

    {
        int x = 10;
        {
            extern int x;
            printf(“x %d\n”, x);
        }
        printf(“x %d\n”, x);
    }
    printf(“x %d\n”, x);

    return 0;
}

static int x = 20;
```

- Compiler error due to linkage disagreement

Advanced C

Storage Classes – External

Example

```
#include <stdio.h>

static int x = 20;

int main()
{
    int x;

    {
        int x = 10;
        {
            extern int x;
            printf("x %d\n", x);
        }
        printf("x %d\n", x);
    }
    printf("x %d\n", x);

    return 0;
}
```

- Compiler error due to linkage disagreement

Advanced C

Storage Classes - External

Example

```
#include <stdio.h>

int x = 20;

int main()
{
    int x;

    {
        int x = 10;
        {
            extern int x;
            printf("x %d\n", x);
        }
        printf("x %d\n", x);
    }
    printf("x %d\n", x);

    return 0;
}
```

- Should be a fine since the compiler refers the same variable x and prints 20