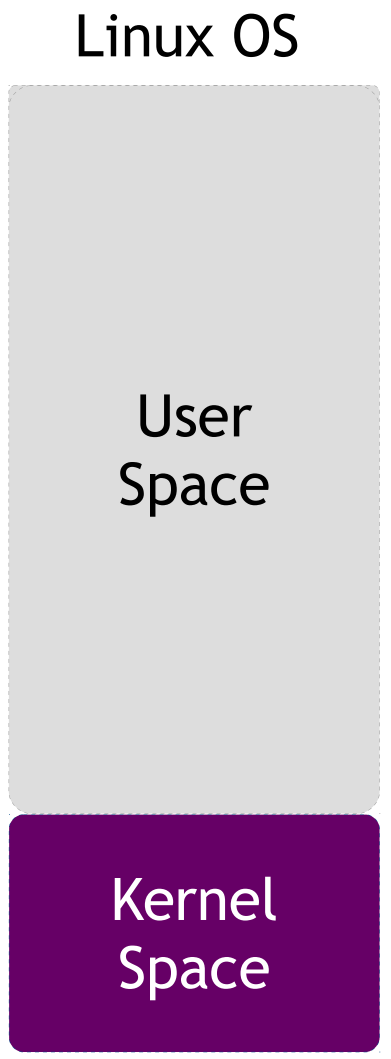
Storage Class

Advanced C

Memory Segments



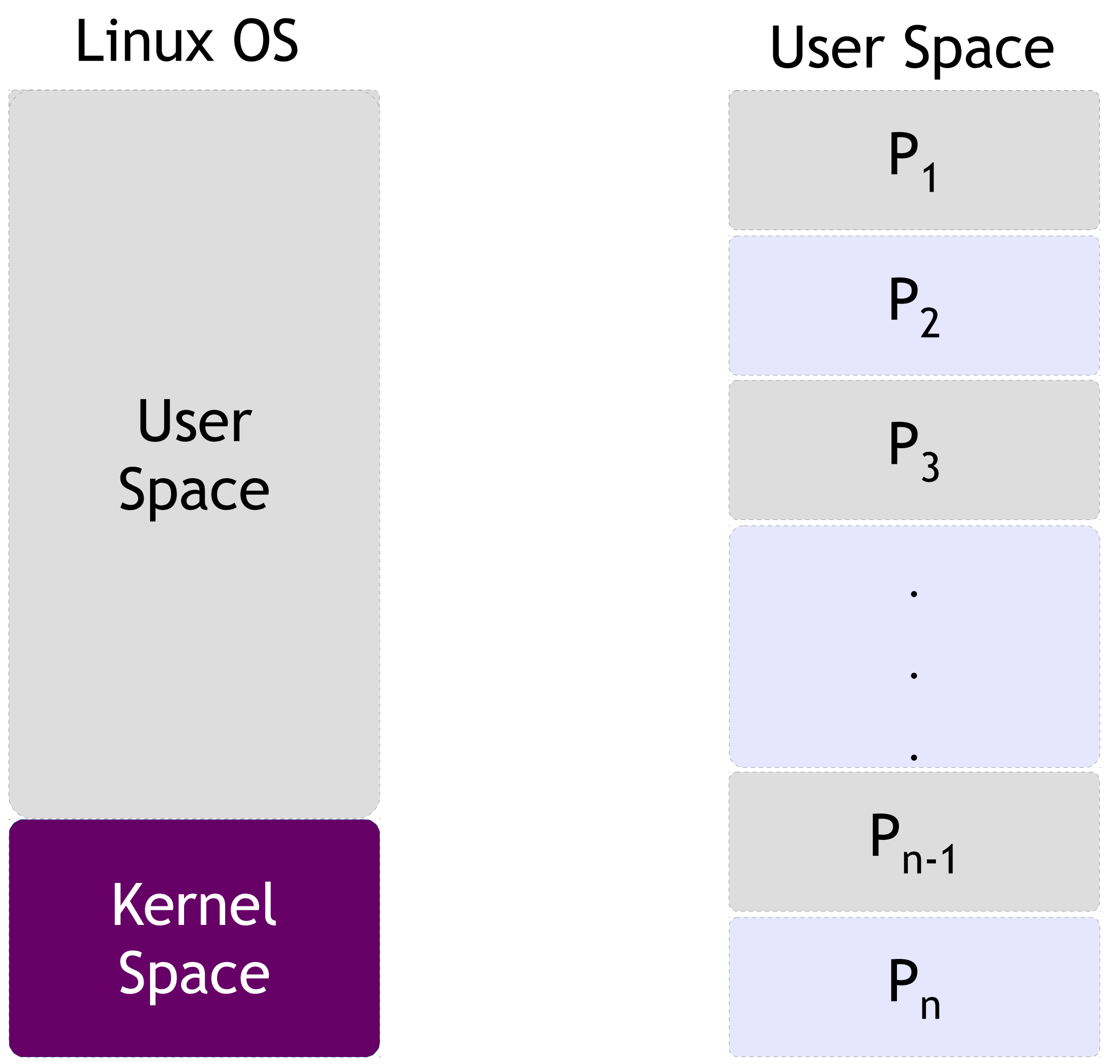
The Linux OS is divided into two major sections
- User Space
- Kernel Space

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



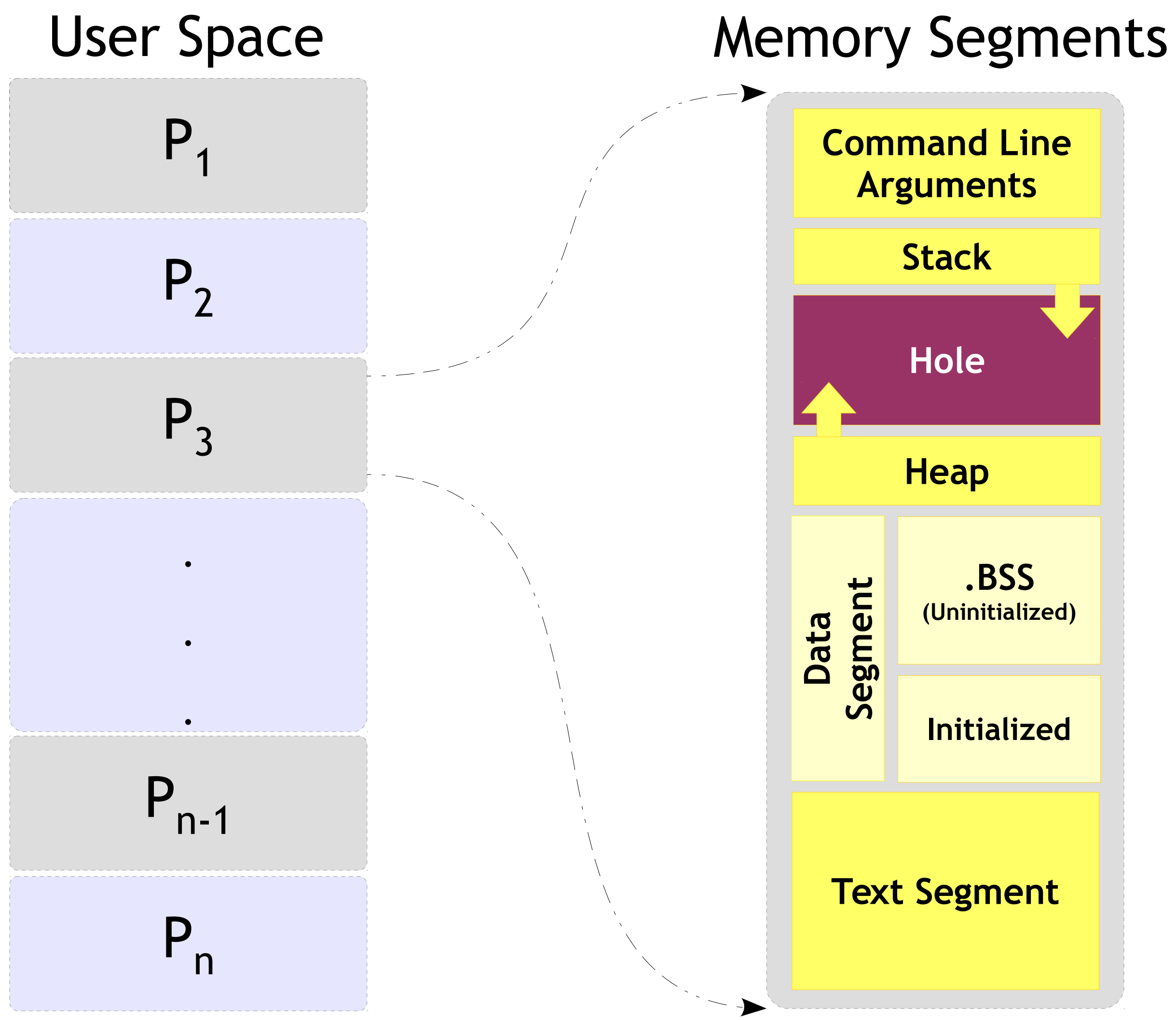
The User space contains many processes

Every process will be scheduled by the kernel

Each process will have its memory layout discussed in next slide

Advanced C

Memory Segments



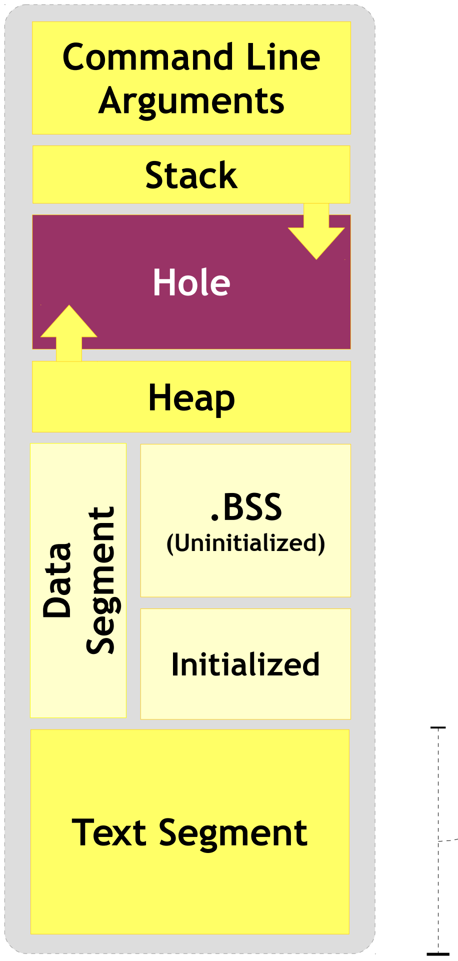
The memory segment of a program contains four major areas.

- Text Segment
- Stack
- Data Segment
- Heap

Advanced C

Memory Segments – Text Segment

Memory Segments

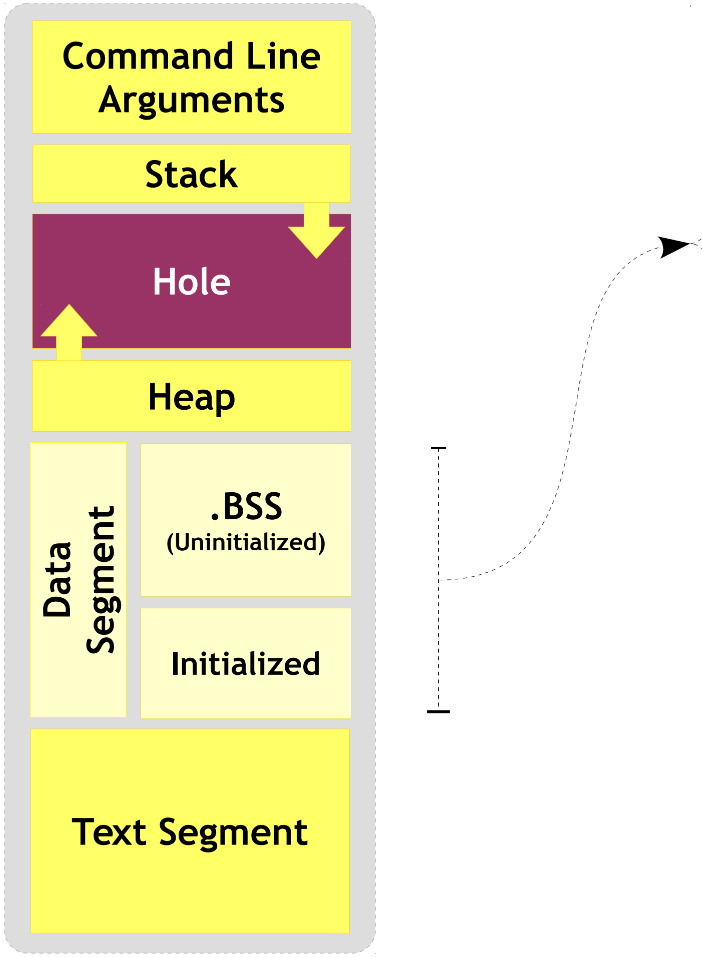


- Also referred as Code Segment
- Holds one of the section of program in object file or memory
- In memory, this is place below the heap or stack to prevent getting over written
- Is a read only section and size is fixed

Advanced C

Memory Segments – Data Segment

Memory Segments



Contains 2 sections as initialized and uninitialized data segments

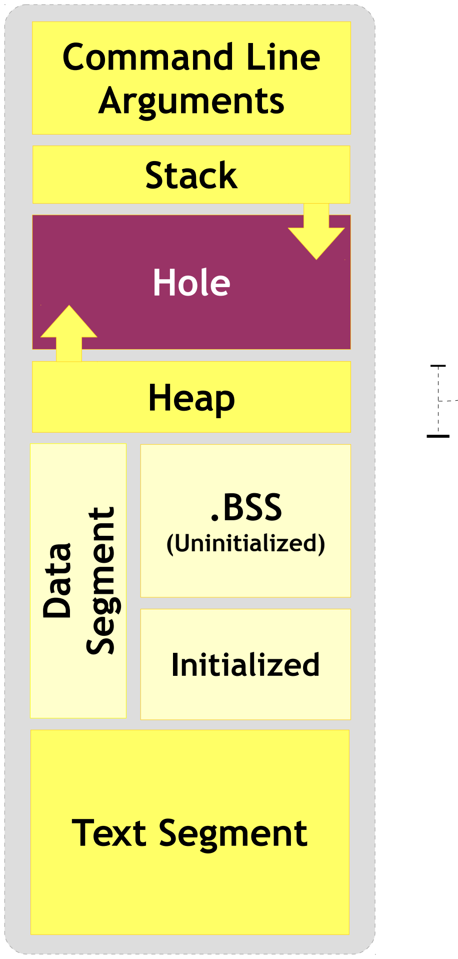
Initialized section is generally called as Data Segment

Uninitialized section is referred as BSS (Block Started by Symbol) usually filled with 0s

Advanced C

Memory Segments – Data Segment

Memory Segments



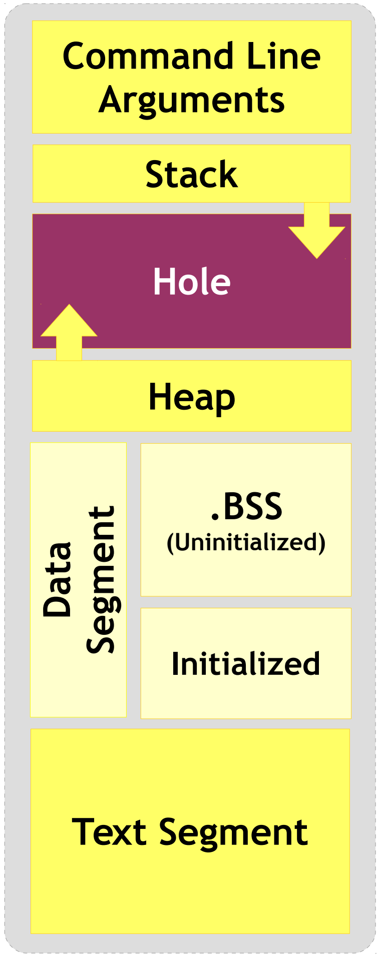
Dynamic memory allocation takes place here

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)

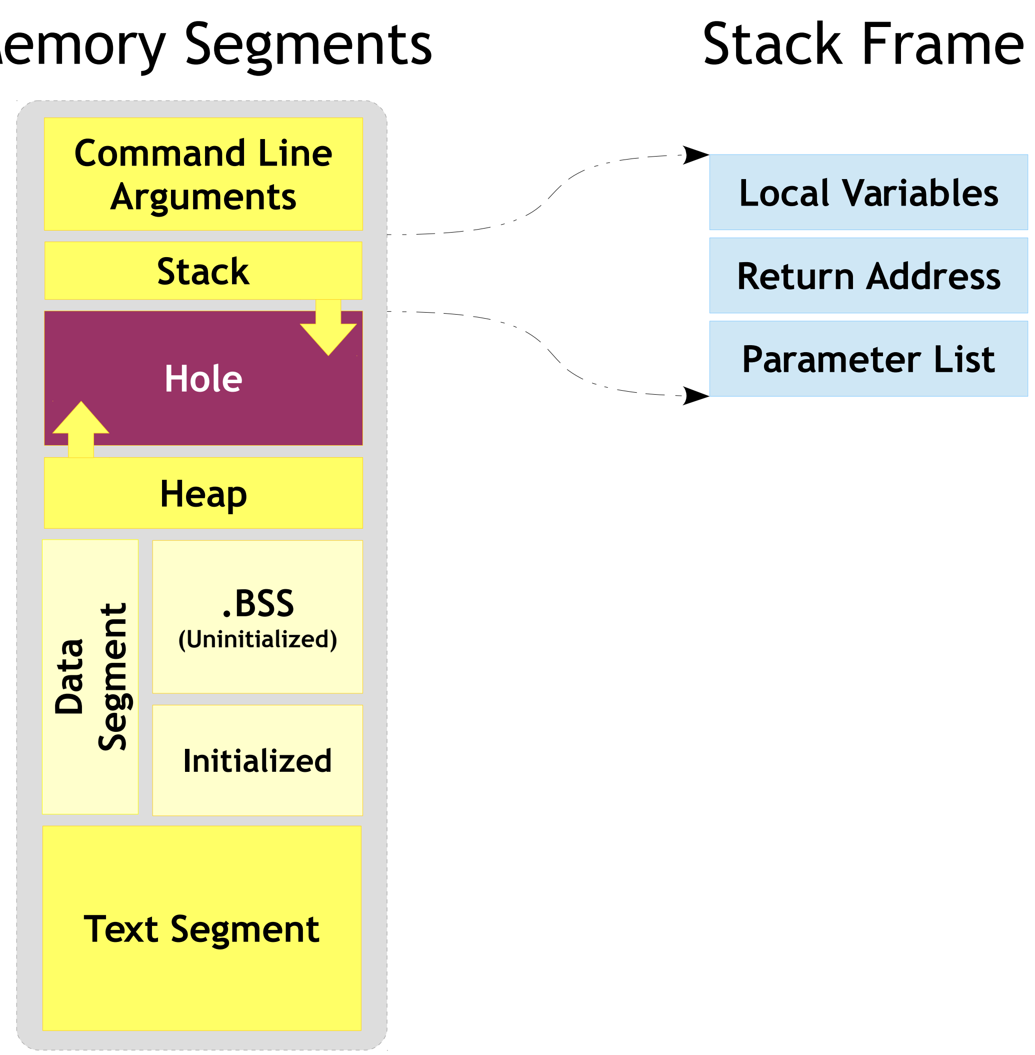
Typically loaded at the higher part of memory

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



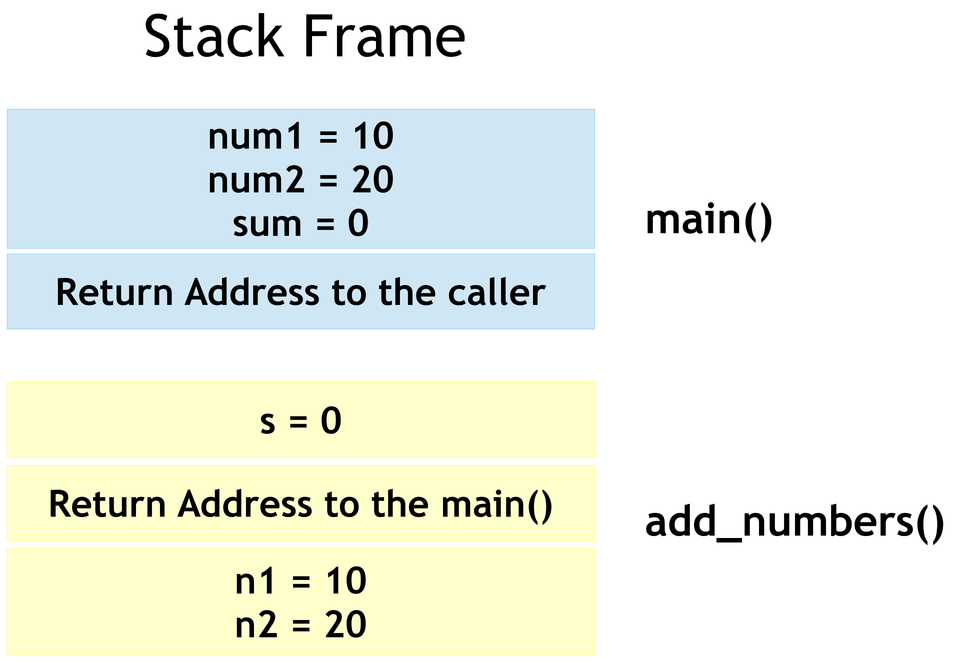
A stack frame contain at least of a return address

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;
}
```



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

Storage Class	Scope	Lifetime	Memory Allocation
auto	Within the block / Function	Till the end of the block / function	Stack
register	Within the block / Function	Till the end of the block / function	Register
static local	Within the block / Function	Till the end of the program	Data Segment
static global	File	Till the end of the program	Data segment
extern	Program	Till the end of the program	Data segment

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

Advanced C

Storage Classes - Auto

Example

```
#include <stdio.h>

int main()
{
    int i = 0;

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

    return 0;
}
```

Advanced C

Storage Classes - Auto

Example

```
#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

Example

```
#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

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

Example

```
#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

Example

```
#include <stdio.h>

int main()
{
    int i = 0;

    {
        int j = 0;

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

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

    return 0;
}
```

EMERTXE

Advanced C

Storage Classes - Auto

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

Example

```
#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

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

Example

```
#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

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

Example

```
#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

Example

```
#include <stdio.h>

char *foo()
{
    static char ca[10] = “Hello World”;

    return ca;
}

int main()
{
    char *ca;

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

    return 0;
}
```

Advanced C

Storage Classes - Static Local

Example

```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

Example

```
#include <stdio.h>

int main()
{
    static int i = 5;

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

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

    return 0;
}
```

Advanced C

Storage Classes - Static Local

Example

```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

Example

```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

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

Example

```
#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

Example

```
#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

Example

```
#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

Example

```
#include <stdio.h>

int x = 10;
int x;

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

    return 0;
}

int main()
{
    foo();

    return 0;
}
```

Advanced C

Storage Classes – Global

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;
}
```

Advanced C

Storage Classes – Static Global

Example

```
#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

Example

```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

file1.c

```c
#include <stdio.h>

int num;

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

    return 0;
}
```

file2.c

```c
#include <stdio.h>

extern int num;

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

    return 0;
}
```

file3.c

```c
#include <stdio.h>

extern int num;

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

    return 0;
}
```

Advanced C

Storage Classes – External

file2.c

```c
#include <stdio.h>

extern int num;
extern int num;

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

    return 0;
}
```

file1.c

```c
#include <stdio.h>

int num;

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

    return 0;
}
```

Advanced C

Storage Classes - External

file2.c

```c
#include <stdio.h>

static int num;
extern int num;

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

    return 0;
}
```

file1.c

```c
#include <stdio.h>

int num;

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

    return 0;
}
```

Advanced C

Storage Classes – External

file2.c

```
#include <stdio.h>

extern char 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;
}
```

Advanced C

Storage Classes - External

file2.c

```
#include <stdio.h>

extern int num;
extern char 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;
}
```

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;
}

int x = 20;
```

Advanced C

Storage Classes – External

Example

```
#include <stdio.h>

int main()
{
    extern char x;

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

    return 0;
}

int x = 0x31;
```

Advanced C

Storage Classes – External

Example

```
#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;
```

Advanced C

Storage Classes – Static Function

file2.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();
}
```

file1.c

```
#include <stdio.h>

int num;

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

    return 0;
}
```

Advanced C

Storage Classes – Static Function

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();
}
```

file1.c

```c
#include <stdio.h>

int num;

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

    return 0;
}
```