Aura Hand BookContents

[**1** **Compilation stages** 3](#_Toc318948524)

[**1.1** **Preprocessing** 3](#_Toc318948525)

[**1.2** **Compilation** 5](#_Toc318948526)

[**1.3** **Linking** 5](#_Toc318948527)

[**1.4** **Loader/Running** 6](#_Toc318948528)

[**2** **Storage Classes and Sections** 6](#_Toc318948529)

[**2.1** **Automatic** 6](#_Toc318948530)

[**2.2** **Extern** 9](#_Toc318948531)

[**2.3** **Static** 12](#_Toc318948532)

[**2.4** **Register** 14](#_Toc318948533)

[**2.5** **Volatile** 14](#_Toc318948534)

[**3** **Scope of variables** 14](#_Toc318948535)

[**3.1** **Local & Global Scope** 14](#_Toc318948536)

[**4** **Pointers** 16](#_Toc318948537)

[**4.1** **Type and value** 16](#_Toc318948538)

[**5** **Strings & Pointers** 17](#_Toc318948539)

[**5.1** **Pointers and expressions** 17](#_Toc318948540)

[**6** **Unions** 19](#_Toc318948541)

[**7** **Unions** 19](#_Toc318948542)

[**8** **Bit manipulations** 19](#_Toc318948543)

[**8.1** **Bit manipulation** 19](#_Toc318948544)

[**9** **Data Types** 19](#_Toc318948545)

[**9.1** **Data Types and Range** 19](#_Toc318948546)

[**10** **Tricky Questions** 19](#_Toc318948547)

[**10.1** **Interview & Tricky Questions** 19](#_Toc318948548)

# **Compilation stages**

## **Preprocessing**

### Macros are replaced after preprocessing

1. c

|  |
| --- |
| #define MY\_NUMBER 5  main()  {  printf("%d\r\n", MY\_NUMBER);  } |

|  |
| --- |
| $ gcc a.c -E  $ gcc a.c -E -o a.i  See the difference between pre1.c and pre1.i  $ gvim -d a.c a.i |

### Conditional compilation

1. c

|  |
| --- |
| #define MY\_NUMBER 5  #define DEBUG  int factorial(int n)  {  int fact = 1;  #ifdef DEBUG  printf("In function n value :%d\r\n", n);  #endif  for (i = 0; i <= n; i++)  {  #ifdef DEBUG  printf("In loop fact and i values :%d - %d\r\n", fact, i);  #endif  fact = fact \* i;  }  #ifdef DEBUG  printf("Before returning from function fact value :%d\r\n", fact);  #endif  return fact;  } |

|  |
| --- |
| main()  {  int fact = 0;  #ifdef DEBUG  printf("Before calling function my\_number :%d\r\n", MY\_NUMBER);  #endif  fact = factorial(MY\_NUMBER);  printf("factorial value %d\r\n", fact);  } |

|  |
| --- |
| $ gcc b.c -o b  $ gcc c.c -o c  See the sizes of b and c files  Execute b and c files and observe the results  $ gcc b.c -E -o b.i  $ gcc c.c -E -o c.i  Remove statement "#define DEBUG" from c.c file  $ gcc c.c -E -o c1.i  $ gcc c.c -o c1  Execute c and c1 executables and see the results  See the difference between c.i and c1.i  $ gcc -DDEBUG -E c3.c -o c2.i  See the difference between c.i and c2.i, c1.i and c2.i  $ gcc -DDEBUG c.c -o c2  Execute the c1 and c2 executables and see the results |

## **Compilation**

1. c

|  |
| --- |
| main()  {  int fact = 10;  printf("fact %d\r\n", fact)  } |

|  |
| --- |
| $ gcc d.c |

## **Linking**

1. c

|  |
| --- |
| #define MY\_NUM 5  main()  {  int retval = 0;  retval = factorial(MY\_NUM);  printf("retval :%d\r\n", retval);  } |

1. c

|  |
| --- |
| main()  {  int fact = 10;  printf("fact %d\r\n", fact)  } |

1. c

|  |
| --- |
| main()  {  int fact = 10;  printf("fact %d\r\n", fact)  } |

## **Loader/Running**

# **Storage Classes and Sections**

## **Automatic**

### What is automatic variable?

### How automatic variable is different from others?

### What is the difference of the output of below programs?

|  |  |
| --- | --- |
| main()  {  auto int a = 10;  auto int b;  printf("-->a :%d\r\n", a);  printf("-->b :%d\r\n", b);  } | main()  {  int a = 10;  int b;  printf("-->a :%d\r\n", a);  printf("-->b :%d\r\n", b);  } |

### What is the output of the below program?

|  |  |
| --- | --- |
| "main.c"  main()  {  auto int a = 10;  printf("-->a :%d\r\n", a);  my\_function1();  my\_function1();  } | "fun1.c"  void my\_function1(void)  {  int temp = 10;  printf("-->temp :%d\r\n", temp);  temp++;  } |

### Does below program compiles? If not, why?

### At what stage compilation gets stopped?

|  |
| --- |
| auto int a = 10;  auto int b;  main()  {  printf("-->a :%d\r\n", a);  printf("-->b :%d\r\n", b);  } |

### Compile below two programs together 'gcc main.c fun1.c"

1. Do they compile? If not, why?
2. At what stage 'main.c' stops compilation? And why?
3. At what stage 'fun1.c' stops compilation? And why?

|  |  |
| --- | --- |
| “main.c”  int a = 10;  int b;  main()  {  printf("-->a :%d\r\n", a);  printf("-->b :%d\r\n", b);  my\_function1();  } | "fun1.c"  void my\_function1(void)  {  printf("-->a :%d\r\n", a);  printf("-->b :%d\r\n", b);  } |

### Compile below two programs together 'gcc main.c fun1.c"

1. Do they compile? If not, why?
2. At what stage 'main.c' stops compilation? And why?
3. At what stage 'fun1.c' stops compilation? And why?

|  |  |
| --- | --- |
| main.c  ------  auto int a = 10;  int b;  main()  {  printf("-->a :%d\r\n", a);  printf("-->b :%d\r\n", b);  my\_function1();  } | "fun1.c"  void my\_function1(void)  {  printf("-->a :%d\r\n", a);  printf("-->b :%d\r\n", b);  } |

### Compile below two programs together 'gcc main.c fun1.c"

1. Do they compile? If not, why?
2. At what stage 'main.c' stops compilation? Why?
3. At what stage 'fun1.c' stops compilation? Why?
4. How to fix the compilation error? And explain the reason?

|  |  |
| --- | --- |
| defs.h  auto int a = 10;  #define NUMBER 5 | main.c  main()  {  printf("-->MAX :%d\r\n", MAX);  printf("-->a :%d\r\n", a);  my\_function1();  } |
| "fun1.c"  #include "defs.h"  void my\_function1(void)  {  printf("-->MAX :%d\r\n", MAX);  printf("-->a :%d\r\n", a);  } |  |

### Compile below two programs together 'gcc main.c fun1.c"

1. Do they compile? If not, why?
2. At what stage 'main.c' stops compilation? And why?
3. At what stage 'fun1.c' stops compilation? And why?
4. How to fix the compilation error? And explain the reason?

|  |  |
| --- | --- |
| "main.c"  int a = 10;  main()  {  printf("-->a :%d\r\n", a);  my\_function1();  } | "fun1.c"  int a = 20;  void my\_function1(void)  {  printf("-->a :%d\r\n", a);  } |

### What is the problem with below program?

### NOTE/Clue: Nothing related with compilation

|  |
| --- |
| #define MY\_NUMBER 5  int \* factorial(int n)  {  int fact = 1;  for (i = 1; i <= n; i++)  {  fact = fact \* i;  }  return &fact;  } |

|  |
| --- |
| main()  {  int \*pfact = 0;  pfact = factorial(MY\_NUMBER);  printf("factorial value %d\r\n", \*pfact);  } |

## **Extern**

### What is extern variable?

### How extern variable is different from normal variable?

### When do we declare the variable as 'extern'?

### Compile below two programs together 'gcc main.c fun1.c"

### Do they compile? If not, why?

|  |  |
| --- | --- |
| "main.c"  int a = 10;  main()  {  printf("-->a :%d\r\n", a);  my\_function1();  } | "fun1.c"  extern int a;  void my\_function1(void)  {  printf("-->a :%d\r\n", a);  } |

### Compile below two programs together 'gcc main.c fun1.c"

1. Do they compile? If not, why?
2. At what stage 'main.c' stops compilation? And why?
3. At what stage 'fun1.c' stops compilation? And why?

|  |  |
| --- | --- |
| "main.c"  main()  {  printf("-->a :%d\r\n", a);  my\_function1();  } | "fun1.c"  extern int a;  void my\_function1(void)  {  printf("-->a :%d\r\n", a);  } |

### Compile below two programs together 'gcc main.c fun1.c".

1. Do they compile? If not, why?
2. At what stage 'main.c' stops compilation? And why?
3. At what stage 'fun1.c' stops compilation? And why?

|  |  |
| --- | --- |
| "main.c"  int a = 10;  int b;  main()  {  printf("-->a :%d\r\n", a);  printf("-->b :%d\r\n", b);  my\_function1();  } | "fun1.c"  extern int a;  void my\_function1(void)  {  printf("-->a :%d\r\n", a);  printf("-->b :%d\r\n", b);  } |

### Compile below two programs together 'gcc main.c fun1.c fun2.c".

1. Do they compile? If not, why?
2. At what stage 'main.c' stops compilation? And why?
3. At what stage 'fun1.c' stops compilation? And why?
4. How to fix the compilation error? And explain the reason?

|  |  |
| --- | --- |
| "fun1.c"  #include "defs.h"  extern int b;  void my\_function1(void)  {  printf("-->MAX :%d\r\n", MAX);  printf("-->b :%d\r\n", b);  } | "main.c"  int a = 10;  int b = 20;  #include "defs.h"  main()  {  printf("-->MAX :%d\r\n", MAX);  printf("-->a :%d\r\n", a);  printf("-->b :%d\r\n", b);  my\_function1();  my\_function2();  } |
| defs.h  extern int a;  #define NUMBER 5 | "fun2.c"  #include "defs.h"  extern int b = 20;  void my\_function2(void)  {  printf("-->MAX :%d\r\n", MAX);  printf("-->a :%d\r\n", a);  printf("-->b :%d\r\n", b);  } |

### Compile below two programs together 'gcc main.c fun1.c fun2.c”

1. Do they compile? If not, why?
2. At what stage 'main.c' stops compilation? And why?
3. At what stage 'fun1.c' stops compilation? And why?
4. How to fix the compilation error? And explain the reason?

|  |  |
| --- | --- |
| defs.h  extern int a;  #define NUMBER 5 | "main.c"  int a = 10;  #include "defs.h"  main()  {  printf("-->MAX :%d\r\n", MAX);  printf("-->a :%d\r\n", a);  my\_function1();  my\_function2();  } |
| "fun1.c"  #include "defs.h"  void my\_function1(void)  {  printf("-->MAX :%d\r\n", MAX);  printf("-->a :%d\r\n", a);  } | "fun2.c"  #include "defs.h"  void my\_function2(void)  {  printf("-->MAX :%d\r\n", MAX);  printf("-->a :%d\r\n", a);  } |

### Compile below two programs together 'gcc main.c fun1.c”

1. Do they compile? If not, why?
2. At what stage 'main.c' stops compilation? And why?
3. At what stage 'fun1.c' stops compilation? And why?
4. How to fix the compilation error? And explain the reason?

|  |  |
| --- | --- |
| "main.c"  int a = 10;  main()  {  printf("-->a :%d\r\n", a);  my\_function1();  my\_function2();  } | "fun1.c"  void function1(void)  {  printf("-->a :%d\r\n", a);  } |
| "fun2.c"  void my\_function2(void)  {  printf("-->a :%d\r\n", a);  } |  |

## **Static**

### Compile below two programs together 'gcc main.c fun1.c func2.c".

1. Do they compile? If not, why?
2. At what stage 'main.c' stops compilation? And why?
3. At what stage 'fun1.c' stops compilation? And why?
4. How to fix the compilation error? And explain the reason?

|  |  |
| --- | --- |
| main.c  static a = 10;  main()  {  printf("-->a :%d\r\n", a);  my\_function1();  my\_function2();  } | "fun1.c"  static int a = 20;  void my\_function1(void)  {  printf("-->a :%d\r\n", a);  } |
| "fun2.c"  static int a = 30;  fun2.c  void my\_function2(void)  {  printf("-->a :%d\r\n", a);  } |  |

### Compile below two programs together 'gcc main.c fun1.c fun2.c".

### What is the value of 'a' in all functions?

|  |  |
| --- | --- |
| "main.c"  int a = 10;  main()  {  printf("-->a :%d\r\n", a);  my\_function1();  my\_function2();  printf("-->a :%d\r\n", a);  } | "fun1.c"  static int a = 20;  void my\_function1(void)  {  printf("-->a :%d\r\n", a);  a++;  printf("-->a :%d\r\n", a);  } |
| "fun2.c"  static int a = 30;  void my\_function2(void)  {  printf("-->a :%d\r\n", a);  a++;  printf("-->a :%d\r\n", a);  } |  |

### Compile below two programs together 'gcc main.c fun1.c”.

### What is the output of the below program?

|  |  |
| --- | --- |
| "main.c"  int a = 10;  main()  {  printf("-->a :%d\r\n", a);  my\_function1();  printf("-->a :%d\r\n", a);  my\_function1();  printf("-->a :%d\r\n", a);  } | "fun1.c"  void my\_function1(void)  {  static int a = 20;  printf("-->a :%d\r\n", a);  a++;  printf("-->a :%d\r\n", a);  } |

Questiong1. How come the local variable in 'my\_function1' is retaining its value across the function calls?

### Compile below two programs together 'gcc main.c fun1.c” Does they compile? If not, why?

1. At what stage 'main.c' stops compilation? And why?
2. At what stage 'fun1.c' stops compilation? And why?
3. How to fix the compilation error? And explain the reason?
4. What do we mean 'file scope variable'?

|  |  |
| --- | --- |
| "main.c"  static int a = 10;  main()  {  printf("-->a :%d\r\n", a);  my\_function1();  printf("-->a :%d\r\n", a);  my\_function1();  printf("-->a :%d\r\n", a);  } | "fun1.c"  void my\_function1(void)  {  printf("-->a :%d\r\n", a);  a++;  printf("-->a :%d\r\n", a);  } |

### What does it mean a function declared as 'static'?

### How to declare a 'file scope function'

## **Register**

### What is the 'register' variable? How to declare it?

### How is it different from normal variable and static variables?

### Can we print the address of 'register' variable?

### In what cases can we use 'register' variable?

## **Volatile**

### What is the 'volatile' variable? How to declare it?

### How is it different from normal variable and static variables?

### Can we print the address of 'volatile' variable?

### In what cases can we use 'volatile' variable?

# **Scope of variables**

## **Local & Global Scope**

### Guess what is the output of the below program?

|  |  |
| --- | --- |
| "main.c"  int a = 10;  main()  {  printf("-->a :%d\r\n", a);  my\_function1();  printf("-->a :%d\r\n", a);  my\_function2();  printf("-->a :%d\r\n", a);  my\_function1();  printf("-->a :%d\r\n", a);  new\_fun();  printf("-->a :%d\r\n", a);  my\_function2();  printf("-->a :%d\r\n", a);  new\_fun();  printf("-->a :%d\r\n", a);  a = my\_function3();  printf("-->a :%d\r\n", a);  } | "main.c"  void my\_function1(void)  {  printf("-->a :%d\r\n", a);  a++;  printf("-->a :%d\r\n", a);  }  void my\_function2(void)  {  int a = 20;  printf("-->a :%d\r\n", a);  a++;  printf("-->a :%d\r\n", a);  }  int my\_function3(void)  {  int a = 30;  printf("-->a :%d\r\n", a);  a++;  printf("-->a :%d\r\n", a);  return a;  } |
| "new\_fun.c"  static int a = 50;  int new\_fun()  {  printf("-->a :%d\r\n", a);  a++;  printf("-->a :%d\r\n", a);  } |  |

### Guess what is the output of the below program?

|  |  |
| --- | --- |
| int a = 10;  main()  {  int a = 20;  int i = 5;  printf("-->a :%d\r\n", a);  if (i > 3)  {  int a = 30;  printf("-->a :%d\r\n", a);  a++;  }  printf("-->a :%d\r\n", a);  } |  |

### Define

### Block Scope,

### Function Scope,

### File Scope and

### Global scope variables with examples

### Define 'static function' properties

### When do we define any function as 'static function'?

# **Pointers**

## **Type and value**

### Fill 'value' and 'types' columns for each notation

### NOTE: Assume address of variables a, pa, ppa, ppa are 500, 600, 700 and 800 respectively

|  |  |
| --- | --- |
| int a = 10;  int \*pa = &a;  int \*\*ppa = &pa;  int \*\*\*pppa = &ppa; |  |

|  |  |  |
| --- | --- | --- |
| Notation | Value | Type |
| a |  |  |
| &a |  |  |
| pa |  |  |
| &pa |  |  |
| \*pa |  |  |
| &\*pa |  |  |
| ppa |  |  |
| \*ppa |  |  |
| &\*ppa |  |  |
| \*\*ppa |  |  |
| &\*\*ppa |  |  |
| \*\*\*pppa |  |  |
| &\*\*\*pppa |  |  |
| &pppa |  |  |
| \*pppa |  |  |
| &\*pppa |  |  |
| &\*\*pppa |  |  |

### Print each byte value of 4 bytes for variable 'a', if

**int a = 300;**

### What is the output the below program?

|  |  |
| --- | --- |
| int a = 0xabcd1234;  printf(“%x”, a);  printf(“%x”, \*(short \*)&a);  printf(“%x”, \*(char \*)&a);  printf(“%x”, \*((char \*)&a) + 1);  \*(char \*)&a = 0xcc;  printf(“%x”, a);  \*(short \*)&a = 0xaabb;  printf(“%x”, a); |  |

### What is the output of the below program? If the address of a is 500

|  |  |
| --- | --- |
| int a = 10;  int \*p = &a; |  |

|  |  |  |
| --- | --- | --- |
| Notation | Value | Type |
| \*p |  |  |
| (char \*)p |  |  |
| p + 1 |  |  |
| ((char \*)p) + 1 |  |  |
| (\*(char \*)p) + 1 |  |  |
| \*((char \*)p + 1) |  |  |
| \*p |  |  |
| (char \*)p |  |  |
| p + 1 |  |  |

### Determine Endianness of your machine? 'Little Endian’ or ‘Big Endian'

### Write a program to reverse the Endianness of a variable.

# **Strings & Pointers**

## **Pointers and expressions**

|  |
| --- |
| char name[100] = "Aura Networks";  char \*ptr1 = "Aura Networks"  char \*ptr2;  NOTE: Assume address of name, ptr1 and ptr2 and constant string are 500, 600, 700 and 300 respectively |

### What is the difference between 'name' and 'ptr'

### What are the values in 'name' and 'ptr'

### What is the output of below statements?

### If the address of the strings are 500 and 700

|  |  |  |
| --- | --- | --- |
| Notation | Value | Type |
| sizeof(name) |  |  |
| sizeof(ptr1) |  |  |
| sizeof(ptr2) |  |  |
| name |  |  |
| ptr1 |  |  |
| &name |  |  |
| &ptr1 |  |  |
| &name+1 |  |  |
| &ptr1+1 |  |  |

### Which of below statements valid/Invalid and Why?

|  |  |
| --- | --- |
| name[3] = 'A'  ptr1[3] = 'A' |  |

### What is the storage section of 'name?'

### What is the storage section of 'ptr?'

### What is the storage section of string "Aura Networks"

### What is the output of below statements?

|  |  |
| --- | --- |
| char name1[] = "Aura Networks"  char name2[] = "Embedded Training"  char \*ptr1 = "Aura Networks"  char \*ptr2 = "Embedded Training"  char \*temp; |  |

### Which of below statements valid/Invalid, Why and what is the output?

|  |  |
| --- | --- |
| temp = ptr1;  ptr1 = ptr2;  ptr2 = temp;  printf("%s :", ptr1);  printf("%s :", ptr2);  temp = name1;  name1 = name2;  name2 = temp;  printf("%s :", name1);  printf("%s :", name2); |  |

### What is the output of below statements, if the base of address of array is 500?

|  |  |  |
| --- | --- | --- |
| Notation | Value | Type |
| "Aura Networks"[4] |  |  |
| 5["Aura Networks"] |  |  |
| "Aura Networks" |  |  |

### What is the difference between Pointer to an array and Array of pointers?

### Which of the below programs compiles successfully and at run time what is the output?

|  |  |
| --- | --- |
| char name[126] = “Aura Networks”;  printf(“%s” name);  name++;  printf(“%s” name); | char \*ptr = “Aura Networks”;  printf(“%s” ptr);  ptr++;  printf(“%s” ptr); |

# **Unions**

# **Unions**

### Using Unions print each byte value of integer variable 'a'

|  |  |
| --- | --- |
| int a = 300; |  |

# **Bit manipulations**

## **Bit manipulation**

### Using bitwise operators print each BYTE value of integer variable 'a'

|  |  |
| --- | --- |
| int a = 300; |  |

### Using bitwise operators reverse Endianness of integer variable 'a'

### Rotate ‘’n’ bits

### Find first set

# **Data Types**

## **Data Types and Range**

### What is the output of the following programs?

|  |  |  |
| --- | --- | --- |
| main()  {  char ch1 = 0;  for (ch1 = 0; ch1 < 255; ch1++)  printf("%d %c\n", ch1, ch1);  } | main()  {  unsigned char ch2 = 0;  for (ch2 = 0; ch2 < 255; ch2++)  printf("%d %c\n", ch2, ch2);  } |  |

### What are the minimum decimal values that can store in 1 bit, 2 bits 3 bits, 6 bits and in 8 bits?

### What is the output of the following programs?

### What is the binary value of 10 and -10?

### What is the binary value of -1?

### What is the output of the bellow program?

|  |  |  |
| --- | --- | --- |
| main()  {  char ch1 = 0;  for (ch1 = 0; ch1 < 255; ch1++)  printf("%d %c\n", ch1, ch1);  } |  |  |

# **Tricky Questions**

## **Interview & Tricky Questions**

### What is the output of the program?

|  |  |
| --- | --- |
| int main()  {  fun(10);  }  int fun(int x)  {  if(x>0)  {  fun(x/2);  }  printf("%d\n", x);  } |  |

### What is the output of the program?

|  |  |
| --- | --- |
| #define max(a,b) (a++ + ++b)  int main()  {  printf("%d\n", max(3,3));  } |  |

### What is the output of the program?

|  |  |
| --- | --- |
| #define max(a, b) (a)>(b)?a:b  main()  {  int i = 10,j = 5,k = 0;  k = max(i++,++j);  printf("i = %d j = %d k = %d\n", i, j, k);  } |  |

### What is the output of the program?

|  |  |
| --- | --- |
| main()  {  union{  struct aa  {  int a;  char b;  };  }b;  printf("%d\n",sizeof(b));  } |  |

### What is the output of the program?

|  |  |
| --- | --- |
| int main()  {  char \*p = "BANGALORE";  printf("%c",++(\*(p++)));  } |  |

### What is the output of the program?

|  |  |
| --- | --- |
| int main()  {  printf("%d",printf("NDS BANGALORE"));  } |  |

### What is the output of the program?

|  |  |
| --- | --- |
| int main()  {  int i = 1;  int a[] = {0,1,2,3,4};  a[i] = ++i;  printf("%d\n",a[i]);  } |  |

### What is the output of the program?

|  |  |
| --- | --- |
| int main()  {  char \*str1 = "abcd";  char str2[]="abcd";  printf("%d\n", sizeof(str1));  printf("%d\n", sizeof(str2));  printf("%d\n”, sizeof("abcd"));  } |  |

### What is the output of the program?

|  |  |
| --- | --- |
| int main()  {  static i = 10;  int i = 50;  printf("%d", i);  } |  |

### What is the value of x?

|  |  |
| --- | --- |
| #include "stdio.h"  #define FIRST char\*  typedef char\* SECOND;  int main()  {  FIRST ch1, ch2;  SECOND ch3, ch4;  }  struct em  {  int i;  struct em \*next;  struct em \*prev;  }em;  x = abc.next->next->prev->next.i;  printf("%d", x); |  |