

INTRODUCTION TO PROGRAMMING C AND JULIA

LECTURE 03:PROGRAM EXECUTION; VARIABLES

Dr Ram Prasad Krishnamoorthy

*Associate Professor
School of Computing and Data Science*

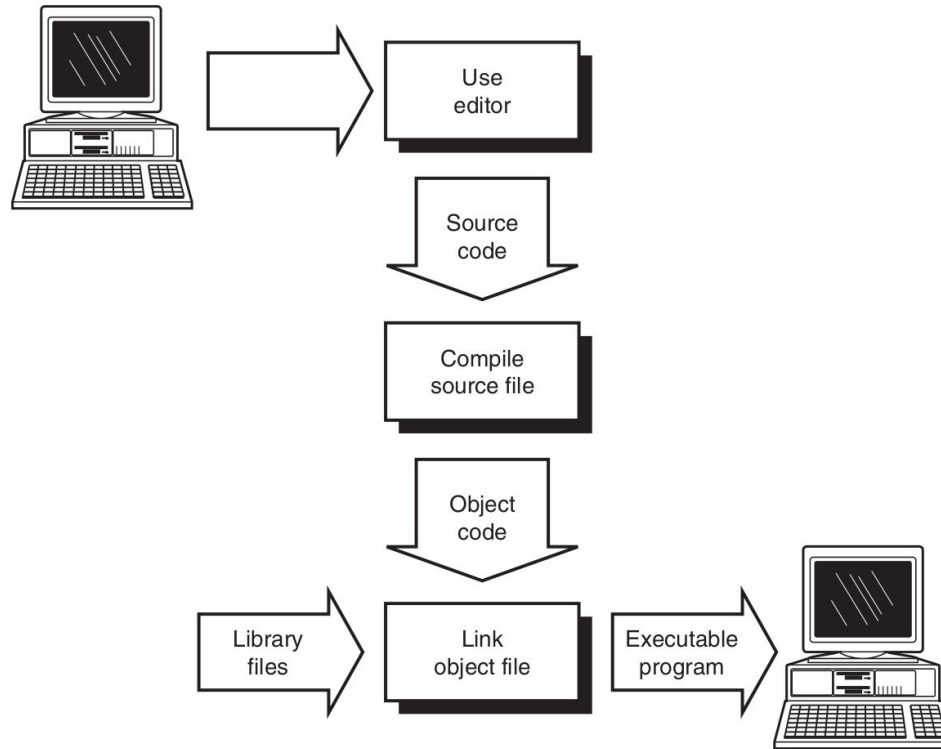
ram.krish@saiuniversity.edu.in



PROGRAM EXECUTION

PROGRAM EXECUTION

**Source code
to
Executable file**



ASCII VS BINARY FILES

ASCII text representation of hello.c

```
#include <stdio.h>

int main()
{
    printf("hello world\n");
    return 0;
}
```

ASCII: American Standard Code for Information Interchange

ASCII text representation of hello.c

#	i	n	c	l	u	d	e	<i>SP</i>	<	s	t	d	i	o	.
35	105	110	99	108	117	100	101	32	60	115	116	100	105	111	46
h	>	\n	\n	i	n	t	<i>SP</i>	m	a	i	n	()	\n	{
104	62	10	10	105	110	116	32	109	97	105	110	40	41	10	123
\n	<i>SP</i>	<i>SP</i>	<i>SP</i>	<i>SP</i>	p	r	i	n	t	f	("	h	e	l
10	32	32	32	32	112	114	105	110	116	102	40	34	104	101	108
l	o	,	<i>SP</i>	w	o	r	l	d	\	n	")	;	\n	<i>SP</i>
108	111	44	32	119	111	114	108	100	92	110	34	41	59	10	32
<i>SP</i>	<i>SP</i>	<i>SP</i>	r	e	t	u	r	n	<i>SP</i>	0	;	\n	}	\n	
32	32	32	114	101	116	117	114	110	32	48	59	10	125	10	

ASCII: American Standard Code for Information Interchange

[illegible]

PROGRAM EXECUTION

hello.c

Source code

<code>#include <stdio.h></code>	→	Preprocessor directive (standard header file)
<code>int main()</code>	→	main() function with no argument
<code>{</code>		
<code>printf("hello world\n");</code>	→	I/O statement
<code>return 0;</code>	→	Control returns back after termination
<code>}</code>		

Compilation in terminal

```
$ gcc hello.c -o hello.bin
```

```
$ ./hello.bin
```

```
hello world
```

Note:

`\n` is a single character (*escape sequence*)

Note:

If we do not specify explicitly the output binary file with **-o** tag, then the default binary executable is named as **a.out**

VARIABLES

PROGRAM VARIABLES

Variables

A name assigned to a location in memory to store data.

Rules for variable names:

- Can contain letters (**a-z**, **A-Z**), digits (**0-9**) and underscore character (**_**)
- Digit **cannot** be used as first character
- C is case sensitive, so **count** and **Count** are two different names
- **Keywords** (*reserved words*, part of C language) cannot be used as variable names
 - There are 32 keywords in C (all of them are **lowercase**)
- **Variables** must be *declared* before it is used in the program

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
continue	for	signed	void
do	if	static	while
default	goto	sizeof	Volatile
const	float	short	Unsigned

<i>Variable Name</i>	<i>Legality</i>
Percent	Legal
y2x5__fg7h	Legal
annual_profit	Legal
_1990_tax	Legal but not advised
savings#account	Illegal: Contains the illegal character #
double	Illegal: Is a C keyword
4sale	Illegal: First character is a digit

PROGRAM VARIABLES

Variables

General syntax:

```
data-type variable-name;
```

Example:

```
int sum;  
float average;  
char alpha;
```

Note:

Always terminate with semicolon (;)

DATA TYPES

C data types and ranges of values

<code>char</code>	Character
<code>int</code>	Integer numbers
<code>float</code>	Real numbers
<code>short</code>	Short integer
<code>long</code>	Long integer
<code>double</code>	Long float

FORMAT SPECIFIERS

FORMAT SPECIFIERS

Placeholder / Format Specifier

<i>Specifier</i>	<i>Meaning</i>	<i>Types Converted</i>
<code>%c</code>	Single character	char
<code>%d</code>	Signed decimal integer	int, short
<code>%ld</code>	Signed long decimal integer	long
<code>%f</code>	Decimal floating-point number	float, double
<code>%s</code>	Character string	char arrays
<code>%u</code>	Unsigned decimal integer	unsigned int, unsigned short
<code>%lu</code>	Unsigned long decimal integer	unsigned long

CALCULATE AVERAGE OF TWO NUMBERS

CALCULATE AVERAGE

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

```
int main(void)  
{
```

main() function with **void** as argument, this is correct as well.

```
/* Finds the average of two values and prints it. */
```

```
int value1 = 15;
```

```
int value2 = 20;
```

```
float average = 0;
```

```
average = (value1 + value2)/2;
```

```
printf("value1: %d, value2: %d, average:%f\n",  
       value1, value2, average);
```

value1: 15, value2: 20, average:17.000000

```
/* Notice the difference when divided by 2 and 2.0 */
```

```
average = (value1 + value2)/2.0;
```

```
printf("value1: %d, value2: %d, average:%f\n",  
       value1, value2, average);
```

value1: 15, value2: 20, average:17.500000

```
return 0;
```

```
}
```

THANK YOU