C99 features in GCC on Fedora

By Shrirang A Kulkarni

The New C

C99 is the C standard ratified by the ANSI and ISO standardizaion groups. It presents a significant amount of changes to the C language. These changes are the result of sibling competition between C and C++. The initial version of C is named after Kernighan and Ritchie. Classic C is K&R C with structure assignment, enumerations, and void. C89, or ANSI C, had some influences from C with Classes. For example, C89 adopted function prototypes in a similar form to what C with Classes provided. The most significant changes in C99, compared to C89, support a variety of new features.

Following is a brief list of C99 features:

- Boolean data types <stdbool.h>
- Increased identifier size limits
- C++ style/line comments
- Inline functions
- Restricted pointers
- Variable Declarations
- Variable length arrays
- New long long type

Specific standards

RECENT POSTS

Fedora 23 End of Life

Endless Sky now available on Fedora

Try Fedora in the cloud for free with Dply

RELATED POSTS

The new C standard for gcc is applied with the following command.

```
$gcc -Wall -std=c99 filename.c
```

Let us consider the following program named *bool.c*:

```
/*bool.c */
#include <stdio.h>
#include <stdbool.h>
int main(int argc, char *argv[])
{
  bool b = true;
  if(b)
    printf("It is a true Boolean data type supported only in C99\n");
  else
   printf("Boolean data type is not supported in C99\n");
  return 0;
}

$ gcc -Wall -std=c99  bool.c
$ ./a.out
It is a true Boolean data type supported only in C99
```

Linking C99 programs with external libraries

The C standard library consists of a collection of headers and library routines used by C programs. The external library is usually stored with an extension *.a* and known as a static library. For instance, the C math

New Firefox 49 features in Fedora

PostgreSQL 9.5: A quick start on Fedora 24

Read about how you can submit an idea or even write an article for Fedora Magazine.

library is typically stored in the file */usr/lib/libm.a* on Linux. The prototype declarations for the functions in the math library are specified in the header file */usr/include/math.h*.

```
/*hypotenuse.c*/
#include <stdio.h>
#include <math.h>
int main(int argc, char *argv[])
{
  float x,y,h;
  //C99 program to demonstrate the use of external math library
function
  printf("\n Enter the values for x and y\n");
  scanf("%f %f", &x,&y);
  h=hypotf(x,y);
  printf(" The Hypotenuse of x and y is %f\n", h);
  return 0;
}
```

Consider the above program *hypotenuse.c*. The following commands are then executed:

```
$ gcc -Wall -std=c99 hypotenuse.c /usr/lib/libm.so -o hypt
$ ./hypt
Enter the values for x and y
6.0
8.0
The Hypotenuse of x and y is 10.000000
```

gcc provides an *-l* option to override long paths when linking against libraries. The following command illustrates this option:

```
$ gcc -Wall -std=c99  hypotenuse.c -lm -o hypt
$ ./hypt
Enter the values for x and y
4.0 8.0
The Hypotenuse of x and y is 8.944272
```

Mixed declarations

ISO C99 allows declarations and code to simultaneously exist in compound statements. The following programming example illustrates this feature:

```
/*mixednewdec.c*/
#include<stdio.h>
int main(int argc, char*argv[])
{
  for(int i=2; i>0 ; --i)
  {
    printf("%d", i);
    int j = i * 2;
    printf("\n %d \n", j);
  }
}

$ gcc -Wall -std=c99  mixednewdec.c
$ ./a.out
2
4
1
2
```

The identifier is visible from where it is declared to the end of the enclosing block.

Variable Length Arrays(VLA)

Variable Length Arrays are not dynamic arrays. Rather, they are created with different sizes each time a declaration is encountered. Only local arrays which are within block scope can be variable arrays.

```
/*vla.c*/
#include<stdio.h>
int main(int argc, char *argv[])
{
  int j = 10;
  void func(int);
  func(j);
  return 0;
}

void func(int x)
{
  int arr[x];
  for(int i=1; i<=x; i++)
  {
    int j=2;
    arr[i] = j*i;
    printf("%d\n",arr[i]);
  }
}
```

Previously array sizes were of fixed size. C99 removes this constraint. It also frees you from performing *allocate( )* and *delete( )* operations on memory explicitly. The output of VLA is illustrated below.

```
$ gcc -Wall -std=c99 vla.c
$ ./a.out
2
4
6
8
10
12
14
16
18
20
```

New Long Long Type

Long long is 64 bit wide integer type. This is the biggest integer type in the C language standard. The long long type was specified to give 32-bit machines a way to handle 64-bit data when interactingwith 64 bit machines. Consider the following C program *longdt.c*:

```
/*longdt.c*/
#include <stdio.h>
int main(int argc, char *argv[])
{
  long long num1 = 123456789101LL;
  long int num2 = 12345678;
  printf("Size of %lld is %u bytes\n", num1 ,sizeof(num1));
  printf("Size of %ld is %u bytes\n", num2 ,sizeof(num2));
  return 0;
}

$ gcc -Wall -std=c99 longdt.c
$ ./a.out
Size of 123456789101 is 8 bytes
Size of 12345678 is 4 bytes
```

Restricted Pointers

C99 lets you prefix pointer declarations with the *restrict* keyword. Thus the pointer itself will be used to access the object it points to. This features takes care of the shortfall of aliasing. It also aids in code optimization. For example, consider the signature of *strcat()* function in the *string.h* file:

```
char *strcat (char* restrict dest,  const char * src)
```

The source string is appended to the end of the destination string. Here, the destination and source strings can be referenced only through the pointers *dest* and *src*. The compiler can then optimize the code generated for the function.

Inline Functions

Inline functions save the overhead of function calls. Consider the following program *inlinecode.c* to demonstrate the use of inline in C99.

```
/*myheader.h*/

#ifndef MYHEADER_H
#define MYHEADER_H

inline int min(int a, int b)
{
  return a < b ? a : b;
}

#endif

/*inlinecode.c*/

#include <stdio.h>
#include "myheader.h"
extern int min(int,int);
int main(int argc, char *argv[])
{
  int a =10, b=20;
  int min_value = min(10,20);
  printf(" The minimum of a and b is %d\n", min_value);
  return 0;
}

$ gcc -Wall -std = c99 inlinecode.c
$ ./a.out
The minimum of a and b is 10
```

Conclusion

C99 is a step ahead in the evolution of ANSI C. It incorporates elegant features like single line comments, boolean data types and larger size data types. C99 also supports code optimization through restricted pointer usage and supports inline functions. Now programmers can exploit these new features and further optimize their code for programming efficiency.

Share:

Email Print Facebook 129 Google Twitter
Reddit Telegram More

Like this:

Like

One blogger likes this.

Shrirang A Kulkarni

December 9, 2016

For Developers

Previous post

6 Comments

ADD YOURS

Frederik
December 9, 2016 at 09:23

C99 has been supported by GCC for ages. The standard setting for GCC was elevated to C99 some years ago (don't recall exactly when). The -new- standard for C is called C11 and has been available since 2011.

The current default C dialect in the current GCC in Fedora is gnu11, meaning C11 with some extensions.

Gary
December 9, 2016 at 12:22

A quick Google search and a wiki read seems to confirm your comment. C99 was released in 1999(big shocker there), while the current standard C11 was released in, again big surprise, 2011.

Gary
December 9, 2016 at 12:23

A quick Google search and a wiki read seems to confirm your comment. C99 was released in 1999(big shocker there), while the current standard C11 was released in, again big surprise, 2011. So not sure what the article is about.

David Novák
December 9, 2016 at 16:26

“New C” 😀

Author is mere 17 years behind… If he wrote about C11, it would still be kinda funny, but acceptable… Writing about C99 (which has been default in GCC for *long* time) is like “eh.. what?”

Smittty
December 10, 2016 at 11:51

Almost nothing mentioned in this article is actually explained in enough detail to be informative, classic example is the inline function example. There is zero examination (proof might be a better word) of *how* the use of inline essentially eliminates the overhead of a function call.

So, not only is this article 17 years too late, but even if it had been produced in 1999, it would be largely uninformative.

My suspicion is that this article was little more than a school assignment so that an H1-B candidate could have something on their resume proving they could speak English and understood basic C programming.

So, nothing to see here, move along.

icywind
December 10, 2016 at 12:36

Is it supposed to be some kind of a joke?

Leave a Reply

Your email address will not be published.

Name

Email

Website

Post Comment

Notify me of follow-up comments by email.

Notify me of new posts by email.

SUBSCRIBE TO FEDORA MAGAZINE VIA EMAIL

Enter your email address to subscribe to this blog and receive notifications of new posts by email.
Join 2,590 other subscribers

Email Address

Subscribe

Read about how you can submit an idea or even write an article for Fedora Magazine.

Search form