Miscellaneous

Advanced C

Miscellaneous - Volatile

- A datatype qualifier
- Most commonly used Embedded Applications
- A keyword instructing the compiler not apply any optimizations on objects qualified with it!
- Why??
  - You know! the compiler sometimes act extra smart on the objects not qualified with volatile
  - Takes implicit assumption the the objects

Advanced C

Miscellaneous - Volatile

001_example.c

```
#include <stdio.h>

int main()
{
     long int wait;
    unsigned char bit = 0;

    while (1)
    {
        bit = !bit;
        printf("The bit is now: %d\r", bit);
        fflush(stdout);
        for (wait = 0xFFFFFFF; wait--; );
    }

     return 0;
}
```

Compile like

```
user@user:~] gcc 001_example.c
```

- Typical embedded bit toggle code
- The output would toggle between 0 and 1 (Depends on the system configuration, tune the delay as required)
- Note the toggle frequency and

Compile like

```
user@user:~] gcc -O3 001_example.c
```

- Now try the same code

ΣMERTXE

Advanced C

Miscellaneous - Volatile

001_example.c

```c
#include <stdio.h>

int main()
{
     volatile long int wait;
    unsigned char bit = 0;

    while (1)
    {
        bit = !bit;
        printf("The bit is now: %d\r", bit);
        fflush(stdout);
        for (wait = 0xFFFFFFF; wait--; );
    }

     return 0;
}
```

Compile like

```
user@user:~] gcc 001_example.c
```

- or

Compile like

```
user@user:~] gcc -O3 001_example.c
```

- Should solve the issue!

- What happens in the previous code is that the compiler see the for loop as an unnecessary code just delaying the system operation
- Hence it removes that statement in the final output
- Adding volatile to the wait restricts the compiler from optimizing the code

EMERTXE

Advanced C

Miscellaneous - Volatile

002_example.c

```c
#include <stdio.h>

int main()
{
    unsigned int i;
    int num;

    for (i = 0; i < 0xFFFFFFFF; i++)
    {
        num = 5;
    }

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

    return 0;
}
```

Compile like

```
user@user:~] gcc 002_example.c
```

- Might take some time to see the output on screen!!

Compile like

```
user@user:~] gcc -O3 002_example.c
```

- Immediate output!!
- Compiler sees the same assignment operation is unnecessarily happening in the loop
- Optimizes the loop, by removing the for loop

Advanced C

Miscellaneous - Volatile

002_example.c

```c
#include <stdio.h>

int main()
{
     volatile unsigned int i;
     int num;

    for (i = 0; i < 0xFFFFFFFF; i++)
    {
         num = 5;
    }

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

    return 0;
}
```

Compile like

```
user@user:~] gcc 002_example.c
```

- or

Compile like

```
user@user:~] gcc -O3 002_example.c
```

- Should solve the issue!
- Both should behave the same way!

Advanced C

Miscellaneous - Volatile

003_example.c

```c
#include <stdio.h>

int main()
{
    int get_out;
    int num = 0;

    scanf("%d", &get_out);

    while (get_out)
    {
        num++;
    }

    return 0;
}
```

Compile like

```
user@user:~] gcc 003_example.c
```

- Enter 1 and should not come out of the loop
- Terminate and run again with input 0, you should not enter the loop

Compile like

```
user@user:~] gcc -O3 003_example.c
```

- Enter 1 and should not enter the loop!! which shouldn't be the case

Advanced C

Miscellaneous - Volatile

003_example.c

```c
#include <stdio.h>

int main()
{
     int get_out;
     volatile int num = 0;

     scanf("%d", &get_out);

     while (get_out)
     {
          num++;
     }

     return 0;
}
```

Compile like

```
user@user:~] gcc 003_example.c
```

- or

Compile like

```
user@user:~] gcc -O3 003_example.c
```

- Should solve the issue!
- Both should behave the same way!

Advanced C

Miscellaneous - Volatile

004_example.c

```c
#include <stdio.h>

int main()
{
    int num1;
    int num2 = 1;

    num1 = ++num2 + num2++ + num2++ + num2++;
    printf("%d\n", num1);

    return 0;
}
```

Compile like

```
user@user:~] gcc 004_example.c
```

- Use the precedence table to obtain the result and verify with output

Run

```
user@user:~] ./a.out
```

Advanced C

Miscellaneous - Volatile

004_example.c

```c
#include <stdio.h>

int main()
{
    int num1;
    volatile int num2 = 1;

    num1 = ++num2 + num2++ + num2++ + num2++;
    printf("%d\n", num1);

    return 0;
}
```

Compile like

```
user@user:~] gcc 004_example.c
```

- and

Run

```
user@user:~] ./a.out
```

- Hmmmh, is it ok now!!