有些programming假设是在compiletime期间就可以下断言的。

```
#include <linux/bug.h>
```

检查n是否时2的多少次方

```
/* Force a compilation error if a constant expression is not a power of 2 */
#define BUILD_BUG_ON_NOT_POWER_OF_2(n)          \
    BUILD_BUG_ON((n) == 0 || (((n) & ((n) - 1)) != 0))
```

比如

```
#define DWC3_TRB_NUM        32
BUILD_BUG_ON_NOT_POWER_OF_2(DWC3_TRB_NUM);

BUILD_BUG_ON_NOT_POWER_OF_2(SCI_MAX_IO_REQUESTS);

BUILD_BUG_ON_NOT_POWER_OF_2(PAGE_SIZE / sizeof(struct hlist_head));
```

zero / NULL则编译报错

```
/* Force a compilation error if condition is true, but also produce a
   result (of value 0 and type size_t), so the expression can be used
   e.g. in a structure initializer (or where-ever else comma expressions
   aren't permitted). */
#define BUILD_BUG_ON_ZERO(e) (sizeof(struct { int:-!!(e); }))
#define BUILD_BUG_ON_NULL(e) ((void *)sizeof(struct { int:-!!(e); }))
```

cond为true则编译报错

```
/**
 * BUILD_BUG_ON_MSG - break compile if a condition is true & emit supplied
 *            error message.
 * @condition: the condition which the compiler should know is false.
 *
 * See BUILD_BUG_ON for description.
 */
#define BUILD_BUG_ON_MSG(cond, msg) compiletime_assert(!(cond), msg)
```

如果编译到了就无条件报错

可以用来check是否编译到了某段code

```
/**
 * BUILD_BUG - break compile if used.
 *
 * If you have some code that you expect the compiler to eliminate at
 * build time, you should use BUILD_BUG to detect if it is
 * unexpectedly used.
 */
#define BUILD_BUG() BUILD_BUG_ON_MSG(1, "BUILD_BUG failed")
```