strlen v1

```
1 size_t strlen_a(const char *str) {
2     size_t length = 0;
3     while (*str++)
4         ++length;
5     return length;
6 }
```

strlen v2

```
size_t strlen_b(const char *str) {
    const char *cp = str;
    while (*cp++)
        ;
    return (cp - str - 1);
}
```

strlen in c standard library

```
    for (char_ptr = str; ((ulong)char_ptr
        & (sizeof(ulong) - 1)) != 0;
        ++char_ptr) {
        if (*char_ptr == '\0')
            return char_ptr - str;
    }

    longword_ptr = (ulong*)char_ptr;
```

The Trick

```
18      magic_bits = 0x7efefeffL;
```

```
                b3       b2        b1        b0
            31--------------------------------->0
magic_bits: 01111110 11111110 11111110 11111111
```

The Purpose: test 0 byte

```
                b3       b2        b1        b0
            31--------------------------------->0
  longword:   XXXXXXXX XXXXXXXX 00000000 XXXXXXXX
+ magic_bits: 01111110 11111110 11111110 11111111
```

The Trick

We find if there exists all 0 byte by

```
23
24          if ((((longword + magic_bits) ^ ~longword) & ~magic_bits) != 0 ) {
```

Get unchanged bits in longword

^ ~longword

Originally longword=01
+magic make it 11,
11 ^ ~ 01 = 11 ^ 10 = 01

& ~magic_bits

Get unchanged “hole”

Results=0 if all unchanged
Otherwise, !=0

The Trick

```
                    b3       b2        b1        b0
                31--------------------------------->0
     longword: 00001001 00011000 00000000 00001100
+  magic_bits: 01111110 11111110 11111110 11111111
           sum: 10001000 00010110 11111111 00001011
^   ~longword: 11110110 11100111 11111111 11110011
            a: 01111110 11110001 00000000 11111000
& ~magic_bits: 10000001 00000001 00000001 00000000
       result: 00000000 00000001 00000000 00000000
```

The Trick

If we find it, compute the offset

```
        if ((((longword + magic_bits) ^ ~longword) & ~magic_bits) != 0 ) {

            const char *cp = (const char*)(longword_ptr - 1 );

            if (cp[0] == 0 )
                return cp -  str;
            if (cp[1] == 0 )
                return cp - str + 1 ;
            if (cp[2] == 0 )
                return cp - str + 2 ;
            if (cp[3] == 0 )
                return cp - str + 3 ;
        }
```

The Trick

If we don't find it, continue the while loop

```
while (1) {

    longword = *longword_ptr++;

    if ((((longword + magic_bits) ^ ~longword) & ~magic_bits) != 0) {

        const char *cp = (const char*)(longword_ptr - 1);

        if (cp[0] == 0)
            return cp - str;
        if (cp[1] == 0)
            return cp - str + 1;
        if (cp[2] == 0)
            return cp - str + 2;
        if (cp[3] == 0)
            return cp - str + 3;
    }
}
```

Why -1?