### 第三十一次2020年7月20日-7月26日

#### 1 关于bell-10无法运行的问题

报错：Segmentation fault(core dumped)

出现这种错误可能的原因（访问了内存中不应该访问的东西）：

1、内存访问越界：

（1）数组访问越界，下标出超出范围。

（2）搜索字符串的时候，通过字符串的结尾符号来判断结束，但是实际上没这个结束符。

（3）使用strcpy, strcat, sprintf, strcmp,strcasecmp等字符串操作数，超出了字符中定义的可以存储的最大范围。使用strncpy, strlcpy, strncat, strlcat, snprintf, strncmp, strncasecmp等函数防止读写越界。

2、多线程程序使用了线程不安全的函数。

3、多线程读写的数据未加锁保护。

对于会被多个线程同时访问的全局数据，应该注意加锁保护，否则很容易造成核心转储

4、非法指针

（1）使用NULL指针

（2）随意使用指针类型强制转换，因为在这种强制转换其实是很不安全的，因为在你不确认这个类型就应该是你转化的类型的时候，这样很容易出错，因为就会按照你强制转换的类型进行访问，这样就有可能访问到不应该访问的内存

5、堆栈溢出

不要使用大的局部变量（因为局部变量都分配在栈上），这样容易造成堆栈

代码问题出现在Load\_Instance()函数中

else

{

long\_watched\_list[big\_clause.lits[0]].push\_back(itr - begin);

lit\_seen[big\_clause.lits[0]] = false;

itr->lits[0] = big\_clause.lits[0];

long\_watched\_list[big\_clause.lits[1]].push\_back(itr - begin);

lit\_seen[big\_clause.lits[1]] = false;

itr->lits[1] = big\_clause.lits[1];

for (i = 2; i < big\_clause.len; i++)

{

long\_watched\_list[big\_clause.lits[i]].push\_back(itr - begin);

lit\_seen[big\_clause.lits[i]] = false;

itr->lits[i] = big\_clause.lits[i];

}

itr->len = big\_clause.len;

}

这段代码是设置文字监听，将文字关联的子句存储到long\_watched\_list[lit]中，此处报错的原因是long\_watched\_list溢出。

long\_watched\_list类型是vector

vector<unsigned> \* long\_watched\_list;

long\_watched\_list的vector大小由SIMPLE\_CNF\_VAR\_BOUND决定

binary\_clauses = new vector<unsigned>[2 \* SIMPLE\_CNF\_VAR\_BOUND + 2];

old\_num\_binary\_clauses = new unsigned[2 \* SIMPLE\_CNF\_VAR\_BOUND + 2];

long\_watched\_list = new vector<unsigned>[2 \* SIMPLE\_CNF\_VAR\_BOUND + 2];

SIMPLE\_CNF\_VAR\_BOUND的大小=10\*64=640，因此之前处理变量数较小的测试用例都没有报错。Bell-10的变量数为1023，造成long\_watched\_list溢出。

#define SIMPLE\_COMPONENT\_CODE\_BITS 10

#define SIMPLE\_CNF\_VAR\_BOUND ( SIMPLE\_COMPONENT\_CODE\_BITS \* ULLONG\_SIZE )

#define SIMPLE\_CNF\_LONG\_CLAUSE\_BOUND ( 2 \* SIMPLE\_COMPONENT\_CODE\_BITS \* ULLONG\_SIZE )

#define ULLONG\_SIZE ( sizeof( unsigned long long ) \* 8 )

如果将SIMPLE\_COMPONENT\_CODE\_BITS这个参数调大，运行就不会报错了。