Mjöllnirzinho Feliz Computing Ltd.

Undefined Behavior em C

Vítor De Araújo
http://inf.ufrgs.br/~vbuaraujo/blog/

Esse tal de C

- Performance > Safety

```
int acessa(int vetor[], int posicao) {
    return vetor[posicao];
}

int main() {
    int vetor[10];
    acessa(vetor, 42);
    return 0;
}
```

- O que acontece?
  - Lê lixo da memória
  - Segmentation fault
  - Terceira guerra mundial

- O que o padrão do C diz?
  - Undefined behavior

Undefined behavior

- Certas ações possuem *comportamento indefinido* em C
- Nesses casos, o programa pode fazer *qualquer coisa*
  - Dar crash
  - Não fazer nada
  - Roubar seu dinheiro e fugir do país
  - i.e., não há *nenhuma garantia* de o que pode acontecer

- Compilador pode fazer o que quiser com comportamento indefinido
  - Contanto que o programa se comporte corretamente nos casos definidos, vale tudo

Think like a compiler – acesso a vetor

```
int acessa(int vetor[], int posicao) {
    return vetor[posicao];
}
```

- Se *posicao* estiver no vetor, basta acessar sem verificar
- Se *posicao* não estiver no vetor, comportamento é indefinido
  - i.e., programa não precisa funcionar nesse caso

- Conclusão: podemos acessar o vetor sem verificar

Think like a compiler – signed overflow

```
#define AVAILABLE_TIME 1000

very_beginning = clock();
deadline = very_beginning + AVAILABLE_TIME;
if (deadline < very_beginning) {
    printf("Overflow!");
}
```

Think like a compiler – signed overflow

```
#define AVAILABLE_TIME 1000

very_beginning = clock();
deadline = very_beginning + AVAILABLE_TIME;
if (deadline < very_beginning) {
    printf("Overflow!");
}
```

- `minimax.c:171:5: warning: assuming signed overflow does not occur when assuming that (X + c) < X is always false [-Wstrict-overflow]`

- i.e., se *c* é positivo, então *X* + *c* só pode ser maior do que *X*
- Isso só não seria verdade se desse overflow
  - Mas overflow em inteiros com sinal é undefined behavior!
    - Logo, compilador pode assumir que não acontece

Think like a compiler – signed overflow

```
#define AVAILABLE_TIME 1000

very_beginning = clock();
deadline = very_beginning + AVAILABLE_TIME;
if (deadline < very_beginning) {
    printf("Overflow!");
}
```

- `minimax.c:171:5: warning: assuming signed overflow does not occur when assuming that (X + c) < X is always false [-Wstrict-overflow]`

- i.e., se *c* é positivo, então *X* + *c* só pode ser maior do que *X*
- Isso só não seria verdade se desse overflow
  - Mas overflow em inteiros com sinal é undefined behavior!
    - Logo, compilador pode assumir que não acontece

Think like a compiler – acesso a ponteiro nulo

```
struct bla_t { int valor; };

int foo(struct bla_t *bla) {
    int valor = bla->valor;
    ...
    if (!bla)
        return ERRO;

    /* faz alguma coisa com valor... */
}
```

Think like a compiler – acesso a ponteiro nulo

```
struct bla_t { int valor; };


int foo(struct bla_t *bla) {
    int valor = bla->valor;
    ...
    if (!bla)
        return ERRO;

    /* faz alguma coisa com valor... */
}
```

Se *bla* fosse nulo, esta linha produziria comportamento indefinido

Think like a compiler – acesso a ponteiro nulo

```
struct bla_t { int valor; };


int foo(struct bla_t *bla) {
    int valor = bla->valor;
    ...
    if (!bla)
        return ERRO;

    /* faz alguma coisa com valor... */
}
```

Se *bla* fosse nulo, esta linha produziria comportamento indefinido

Teste sempre falso (só seria verdadeiro em caso de comportamento indefinido, so we don't care)

Troll compiler?

```
struct bla_t { int valor; };

void principal(struct bla_t *bla){
    if (!bla)
        exit(ERRO);
    foo(bla);
}

int foo(struct bla_t *bla) {
    int valor = bla->valor;
    imprime_estrutura(bla);
    return valor * valor;
}

void imprime_estrutura(struct bla_t *bla) {
    if (!bla)
        printf("Nada\n");
    else
        printf("{ %d }\n", bla->valor);
}
```

Troll compiler?

```
struct bla_t { int valor; };

void principal(struct bla_t *bla){
    if (!bla)
        exit(ERRO);
    foo(bla);
}

int foo(struct bla_t *bla) {
    int valor = bla->valor;
    if (!bla)
        printf("Nada\n");
    else
        printf("{ %d }\n", bla->valor);
    return valor * valor;
}
```

Troll compiler?

```
struct bla_t { int valor; };

void principal(struct bla_t *bla){
    if (!bla)
        exit(ERRO);
    foo(bla);
}

int foo(struct bla_t *bla) {
    int valor = bla->valor;
    if (!bla)
        printf("Nada\n");
    else
        printf("{ %d }\n", bla->valor);
    return valor * valor;
}
```

Conclusão

- Undefined behavior
  - Não quer dizer que o programa vá dar crash (pode ser muito pior)
  - É fonte de vulnerabilidades de segurança
  - Permite otimizações malandras

- Leia os warnings do compilador

- http://blog.regehr.org
- https://lwn.net/Articles/342330/
- http://blog.llvm.org/2011/05/what-every-c-programmer-should-know_14.html

xkcd.com

Conclusão

- Undefined behavior
  - Não quer dizer que o programa vá dar crash (pode ser muito pior)
  - É fonte de vulnerabilidades de segurança
  - Permite otimizações malandras

- Leia os warnings do compilador

- http://blog.regehr.org
- https://lwn.net/Articles/342330/
- http://blog.llvm.org/2011/05/what-every-c-programmer-should-know_14.html

xkcd.com