Otimização em C

História, Mitos &
Esquisitices

Daniel Quadros – dqsoft.blogspot.com – dqsoft.blogspot@gmail.com

O Segredo do Sucesso do C

Despite some aspects mysterious to the beginner and occasionally even to the adept, **C remains a simple and small language, translatable with simple and small compilers**. Its types and operations are well-grounded in those provided by real machines, and for people used to how computers work, **learning the idioms for generating time- and space-efficient programs is not difficult**.

Dennis M. Ritchie
The Development of the C Language

Um exemplo: Register

- Pode ser ignorado
- Implementação define quantos, tipos, etc.
- Programador ajudando o compilador
- Obsoleto para compiladores modernos

Exemplo de Otimização Clássica

```
typedef struct
{
    int codigo;
    double peso;
    double volume;
} OBJETO;
OBJETO tabela[100];
```

```
int i;
for (i = 0; i < 100; i++)
  if (tabela[i].codigo == cod)
    break;
```

```
OBJETO *p;
for (p = tabela;
     p < &tabela[100]; p++)
  if (p->codigo == cod)
    break;
```

Compiladores modernos geram o mesmo código para os dois casos!

Evolução dos Compiladores

Vendor	Overall performance and misc.:						
	looptst	optim	fibtest	sieve	rsieve	psieve	isort
Aztec (Manx)	37.8	35.7	56.2	99.0	58.0	53.3	42.0
Control C	62.2	40.9	63.0	103.4	59.1	54.8	60.0
C Systems	100.8	73.1	69.7	173.8	132.6	190.7	51.5
Computer Innovations	66.1	39.8	54.0	117.5	117.4	119.3	49.0
Datalight	63.0	35.3	49.8	99.9	100.0	101.3	42.6
DeSmet	63.0	46.4	52.9	108.2	108.4	103.4	53.8
Digital Research	58.8	66.9	51.7	104.8	58.8	57.0	49.9
EcoSoft	63.0	50.7	52.0	115.3	115.4	138.1	85.7
Lattice	63.0	25.0	70.0	100.3	100.2	96.9	77.2
Mark Williams	61.8	40.8	63.0	103.3	59.0	54.7	59.7
Microsoft	58.8	59.0	59.0	99.4	54.9	50.0	44.3
Software Toolworks	74.2	31.8	56.4	133.6	110.3	176.8	123.6
Wizard	58.8	68.8	52.6	101.0	55.7	51.1	60.0

Table 3a
Execution Times

Evolução dos Compiladores

- Microsoft C (16 bits)
  - Versões 1 e 2: Lattice
  - Versão 3: Primeira versão Microsoft
  - Versão 4: Otimização, CodeView
  - Versão 5.1: “Microsoft C 5.1 Optimizing Compiler”

Quanto Tempo Demora?

- 8080
  - INR r1 5 ciclos
  - INR M 10 ciclos
  - JZ addr 10 ciclos

Quanto Tempo Demora?

- 8086
  - INC reg8 3 ciclos
  - INC reg16 2 ciclos
  - INC M 15+EA ciclos
  - MUL reg16 118-133 ciclos
  - JZ addr 4 ou 16 ciclos
  - Tempos de execução, após fetch e decode

Quanto Tempo Demora?

- Microprocessadores modernos (Pentium...)
  - Caches
  - Micro-ops
  - Branch predictor, speculative execution, out-of-order completion, register renaming
  - ...

Algumas Otimizações Válidas

- Substituir ponto flutuante por inteiro (por enquanto...)
- Escolher a ordem em expressões booleanas
  - Ex: If ((a < 10) && (b > 40))
- Usar if ao invés switch se alguns casos são muito mais prováveis que outros
- Minimizar chamadas a rotinas externas

Ponteiro: Inimigo da Otimização

```
int a = 2;
int func (int *p)
{
  *p = 2*a;
  return 2*a;
}

 func (&a);
```

Mitos

- Código otimizado é mais complicado
- Otimizar requer violar as boas práticas
- Generalizações de compiladores/ambientes específicos
  - Use o menor tipo numérico possível
  - Use ponteiros ao invés de índices
  - Usar unsigned ao invés de signed

Otimizando Demais

- Otimização pode
  - Mudar a ordem
  - Eliminar código
- Exemplos de problema
  - Loops de temporização
  - Acessos a I/O via memória

Otimizando Demais

Undefined Behavior

“ However, if any such execution contains an undefined operation, this International Standard places no requirement on the implementation executing that program with that input (not even with regard to operations preceding the first undefined operation).”

Otimizando Demais

- Exemplos de “Undefined Behavior”
  - Usar variável não iniciada
  - Acessar (ler) memória não alocada
  - Overflow em inteiro com sinal
  - Shift de “1” no bit de sinal (C99)

Otimizando Demais

```
int table[4];
bool exists_in_table(int v)
{
    for (int i = 0; i <= 4; i++) {
        if (table[i] == v) return true;
    }
    return false;
}
```

Otimizando Demais

```
int table[4];
bool exists_in_table(int v)
{
  if (
    re
  if (
    re
  if (
    re
  if (
    return true;
  return true;
}
```

```
int table[4];
bool exists_in_table(int v)
{
    return true;
}
```

Como Otimizar

- Não seja um otimizador precoce
- Não incorra em ineficiências grosseiras
- Verifique onde está o gargalo
- Mudar algoritmo e/ou estrutura de dados pode dar ganho maior que otimizar código
- Confie no seu compilador
- Baseie suas otimizações nas coisas que o compilador não tem como saber

Referências

- The Development of the C Language
  - http://cm.bell-labs.com/who/dmr/chist.html
- Undefined Behavior x Otimização
  - blogs.msdn.com/b/oldnewthing/archive/2014/06/27/10537746.aspx
  - http://blog.regehr.org/archives/759
- Cursos no Coursera
  - https://www.coursera.org/course/algo
  - https://www.coursera.org/course/algo2
  - https://www.coursera.org/course/optimization

Participe do

Garoa Hacker Clube