Duff's device

c언어 스위치문으로 할 수 있는 것

성우경

2021.12.01

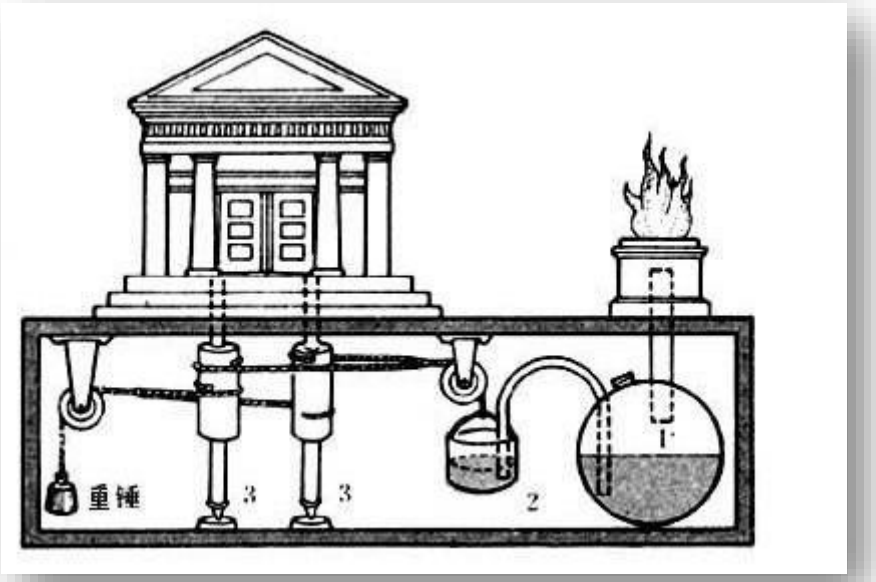
1 이번 시간은

유명하지 않지만 재미있는 C언어 기능 소개

업무에는 그다지 필요 없음

2 더프의 장치

이런 게 상상 되지만 아닙니다



3 더프의 장치

루카스필름, 픽사에 근무했던 톰 더프가 발견한 최적화

4 흔한 복사 함수

문자열이나 메모리를 복사하는 코드

```
void my_copy(char* from, char* to, int count)
{
    while (count--) {
        *to++ = *from++;
    }
}
```

5 더프의 장치

스위치 문법의 헐거움을 이용한 루프풀기 최적화

이유는 모르겠지만 메모리 복사가 빠르다

```
int duffs_device(char* from, char* to, int count)
{
    int n = (count + 7) / 8;

    switch (count % 8) {
    case 0:
        while (n-- > 0) {
        *to++ = *from++;
    case 7:
        *to++ = *from++;
    case 6:
        *to++ = *from++;
    case 5:
        *to++ = *from++;
    case 4:
        *to++ = *from++;
    case 3:
        *to++ = *from++;
    case 2:
        *to++ = *from++;
    case 1:
        *to++ = *from++;
        }
    }

    return count;
}
```

6 더프의 장치

이 괴상한 코드는 유효한 문법

스위치문 case label은 goto 문 label 과 비슷

그래서 스위치문 case label 은 위치 제약 없음

```
int duffs_device(char* from, char* to, int count)
{
    int n = (count + 7) / 8;

    switch (count % 8) {
    case 0:
        while (n-- > 0) {
            *to++ = *from++;
    case 7:
            *to++ = *from++;
    case 6:
            *to++ = *from++;
    case 5:
            *to++ = *from++;
    case 4:
            *to++ = *from++;
    case 3:
            *to++ = *from++;
    case 2:
            *to++ = *from++;
    case 1:
            *to++ = *from++;
        }
    }

    return count;
}
```

switch 문
while 문

7 더프의 장치 실행 속도

이 괴상한 코드는 꽤 빠르다

256 MB 의 메모리를 복사하는데 걸린 시간을 측정하였으며, i5 + 4GB ram + Windows7 64bit 머신에서 Release 로 빌드한 결과이다.

종류	시간 (단위:ms)
일반 복사	347.66
Duff's Device 복사	174.86

<출처: http://z3moon.com/프로그래밍/duff_s_device>

8 더프의 장치 원리

반복문은 반복할 때마다 비교, 분기 연산이 발생
때문에 반복을 줄이는 최적화가 존재
= 루프풀기(Loop unrolling)

배열 복사 단위가 8바이트 배수라면
반복을 1/8 로 줄일 수 있다
(오른쪽 코드 참고)

하지만 8의 배수가 아닌 경우는 난감

```
void my_copy8x(char* from, char* to, int count)
{
    int n = count / 8;

    while (n-- > 0) {
        *to++ = *from++;
        *to++ = *from++;
        *to++ = *from++;
        *to++ = *from++;
        *to++ = *from++;
        *to++ = *from++;
        *to++ = *from++;
        *to++ = *from++;
    }
}
```

9 더프의 장치 원리

더프의 장치는 스위치문 case label 의 위치 제약이 없는 점을 활용

루프풀기와 범용성 둘다 만족

```
int duffs_device(char* from, char* to, int count)
{
    int n = (count + 7) / 8;

    switch (count % 8) {
    case 0:
        while (n-- > 0) {
        *to++ = *from++;
    case 7:
        *to++ = *from++;
    case 6:
        *to++ = *from++;
    case 5:
        *to++ = *from++;
    case 4:
        *to++ = *from++;
    case 3:
        *to++ = *from++;
    case 2:
        *to++ = *from++;
    case 1:
        *to++ = *from++;
        }
    }

    return count;
}
```

10 효용성

지금은 유용하지 않다

40년 동안 컴파일 최적화 기능과
CPU 아키텍쳐가 발전

괴수들의 손을 거쳐간 memcpy 함수가
가장 빠름 (오른쪽 어셈블러 참고)

C언어 잠재력을 찾을 수 있는 점에 가치

/ arch / arm64 / lib / memcpy.S

```
58  */
59
60  SYM_FUNC_START_ALIAS(__memmove)
61  SYM_FUNC_START_WEAK_ALIAS_PI(memmove)
62  SYM_FUNC_START_ALIAS(__memcpy)
63  SYM_FUNC_START_WEAK_PI(memcpy)
64          add     srcend, src, count
65          add     dstend, dstin, count
66          cmp     count, 128
67          b.hi    L(copy_long)
68          cmp     count, 32
69          b.hi    L(copy32_128)
70
71          /* Small copies: 0..32 bytes.  */
72          cmp     count, 16
73          b.lo    L(copy16)
74          ldp     A_l, A_h, [src]
75          ldp     D_l, D_h, [srcend, -16]
76          stp     A_l, A_h, [dstin]
77          stp     D_l, D_h, [dstend, -16]
78          ret
79
80          /* Copy 8-15 bytes.  */
81  L(copy16):
82          tbz     count, 3, L(copy8)
83          ldr     A_l, [src]
84          ldr     A_h, [srcend, -8]
85          str     A_l, [dstin]
86          str     A_h, [dstend, -8]
87          ret
88
89          .p2align 3
90          /* Copy 4-7 bytes.  */
91  L(copy8):
92          tbz     count, 2, L(copy4)
93          ldr     A_lw, [src]
94          ldr     B_lw, [srcend, -4]
95          str     A_lw, [dstin]
96          str     B_lw, [dstend, -4]
97          ret
```

감사합니다