restrict Gösterici Nedir ve Ne İşe Yarar?

15-Ağustos-2009

restrict gösterici kavramı resmi olarak *C99* ile standartlara girdi. *C++* standardizasyon komitesi *restrict* göstericilere hiç sıcak bakmadı. *C++0x*'te de *restrict* göstericilerin eklenmesine yönelik bir işaret yok.

restrict göstericiler aynı nesneyi gösteren göstericilerin (*pointer aliasing*) yol açtığı optimizasyon engelini ortadan kaldırmak için düşünülmüştür. Aynı nesneyi gösteren göstericilere yönelik optimizasyonları "*Derleyicilerin Kod Optimizasyonları 2. Bölüm"* başlıklı makalede açıklamıştım. Önce kısa bir tekrar yapayım.

Göstericinin gösterdiği yere erişmek pek çok işlemcide birden fazla makina komutunu gerektirmektedir. Bu nedenle derleyiciler optimizasyon amacıyla her defasında göstericinin gösterdiği yere erişmek yerine göstericinin gösterdiği yerdeki nesneyi geçici bir yerde tutup oradan kullanmak isteyebilirler. Fakat derleyicilerin böyle bir optimizasyonu yapabilmesi için göstericinin gösterdiği yerdeki nesnenin başka bir yolla değişmeyeceğinden emin olmaları gerekir. Örneğin:

```
void foo(int *p)
{
    int i;

    for (i = 0; i < 100; ++i)
        bar(*p * 2);
}
```

burada derleyici **p*'yi bir yerde saklayıp onu kullanarak hız kazancı sağlayabilir mi? Örneğin bu kod aşağıdaki ile eşdeğer midir?

```
void foo(int *p)
{
    int i;

    int temp = *p;
    for (i = 0; i < 100; ++i)
        bar(temp * 2);
}
```

Hayır. Çünkü *p* global bir nesneyi gösteriyor olabilir ve *bar* fonksiyonu da bu nesneyi değiştiriyor olabilir. Derleyici böyle bir olasılığı göz ardı edemez. Peki derleyici *bar* fonksiyonunun *p*'nin gösterdiği yerdeki nesneyi değiştirmediğinden nasıl emin olabilir? Eğer derleyici *bar* fonksiyonunun kodlarını görürse (başka bir modüldeyse göremeyebilir) bunu anlayabilir. Tabi derleyici için bunun hiç de kolay olmadığını söylemeliyim. Şimdi blok kopyalaması yapan *memcpy* benzeri bir fonksiyon yazmak isteyelim. Böyle bir fonksiyonun klasik yazımı aşağıdaki gibidir:

```
void *copymem(void *dest, const void *source, size_t size)
{
    char *cdest = (char *) dest;
    const char *csource = (const char *) source;

    while (size-- > 0)
        *cdest++ = *csource++;

    return dest;
}
```

Derleyicinin yapılmak istenen şeyi farkettiğini varsayalım. Kopyalamayı *byte byte* değil de *long long* yaparak hız kazancı elde etmek istesin. Bu kodu aşağıdaki gibi derleyebilir mi?

```
void *copymem(void *dest, const void *source, size_t size)
{
    char *cdest = (char *) dest;
    const char *csource = (const char *) source;
    size_t count, remainder;

    count = size / sizeof(long);
    remainder = size % sizeof(long);

    while (count-- > 0) {
        *(long *) cdest = *(const long *) csource;
        csource += sizeof(long);
        cdest += sizeof(long);
    }
    while (remainder-- > 0)
        *cdest++ = *csource++;

    return dest;
}
```

Hayır. Çünkü bu iki kod eşdeğer değildir. Bloklar çakışmasa tamam ama ya çakışırsa?.. Örneğin:

```
copymem(p + 1, p, 100);
```

gibi bir çağrıda birinci fonksiyonla ikinci fonksiyonun davranışı tamamen farklıdır.

İşte *restrict* göstericiler bir nesneye o göstericiden başka bir yolla erişilmeyeceği sözünü derleyiciye veriyorlar. Yani bir göstericiyi *restrict* yapmakla siz derleyiciye o göstericinin gösterdiği yerlere yalnızca o gösterici yoluyla erişileceğinin garantisini veriyorsunuz[1]. Şimdi yeniden birinci örneğe dönelim:

```
void foo(int * restrict p)
{
    int i;

    for (i = 0; i < 100; ++i)
        bar(*p * 2);
}
```

Burada derleyici artık *p* göstericisinin gösterdiği yerin başka bir biçimde değiştirilmeyeceğinden emin olur ve kodu aşağıdaki gibi iyileştirebilir:

```
void foo(int * restrict p)
{
    int i;

    int temp = *p;
    for (i = 0; i < 100; ++i)
        bar(temp * 2);
}
```

Şimdi de *copymem* örneğine dönelim:

```
void *copymem(void * restrict dest, const void * restrict
source, size_t size)
{
    char *cdest = (char *) dest;
    const char *csource = (const char *) source;

    while (size-- > 0)
        *cdest++ = *csource++;

    return dest;
}
```

Siz burada derleyiciye *dest* ve *source* göstericileriyle eriştiğiniz yerlere başka bir yolla erişmeyeceğiniz sözünü vermiş oluyorsunuz. O halde bu durumda bloklar da çakışık olamaz. Çünkü blokların çakışık olması onların gösterdiği yerlere başka biçimlerde erişebileceğiniz anlamına gelir. İşte derleyici bu garanti altında kodu optimize edebilir.

restrict niteleyicisi yalnızca göstericiler için ve yalnızca göstericilerin kendisi için kullanılabilir. Göstericilerin gösterdiği yer *restrict* olamaz. Örneğin:

```
restrict int a;          // geçersiz!
restrict char *p2;       // geçersiz!
char * restrict p1;      // geçerli
```

restrict bir göstericinin yaşamı boyunca o gösterici ile erişilen nesnelere başka bir yolla erişilmemelidir. Eğer erişilirse tanımsız davranış (*undefined behavior*) oluşur. Böylece örneğin:

```
void *copymem(void * restrict dest,
              const void * restrict source, size_t size);
```

biçiminde bildirilmiş bir fonksiyona biz çakışık blok adresi geçemeyiz. Eğer geçersek programımız sağlıklı çalışmayabilir.

C99'da pek çok standart *C* fonksiyonunun prototipi değiştirilerek göstericiler *restrict* yapılmıştır. Göstericileri *restrict* yapılan bazı standart *C99* fonksiyonları şunlardır:

```
FILE *fopen(const char * restrict filename,
                const char * restrict mode);

int fprintf(FILE * restrict stream,
                const char * restrict format, ...);

int fscanf(FILE * restrict stream,
                const char * restrict format, ...);

int printf(const char * restrict format, ...);

int scanf(const char * restrict format, ...);

int snprintf(char * restrict s, size_t n,
                const char * restrict format, ...);

int sprintf(char * restrict s,
                const char * restrict format, ...);

int sscanf(const char * restrict s,
                const char * restrict format, ...);

char *fgets(char * restrict s, int n, FILE * restrict stream);

int fputs(const char * restrict s, FILE * restrict stream);

size_t fread(void * restrict ptr, size_t size,
                size_t nmemb, FILE * restrict stream);

size_t fwrite(const void * restrict ptr, size_t size,
                size_t nmemb, FILE * restrict stream);

int fgetpos(FILE * restrict stream, fpos_t * restrict pos);

void *memcpy(void * restrict s1,
                const void * restrict s2, size_t n);

char *strncpy(char * restrict s1,
                const char * restrict s2, size_t n);

char *strcat(char * restrict s1, const char * restrict s2);
```

```
char *strtok(char * restrict s1, const char * restrict s2);
```

---

[1] Tabi *restrict* bir gösterici kullandığınızda o göstericiyle erişebilme hakkınızın olduğu tüm yerlere başka bir biçimde erişmeme sözü veriyorsunuz. Örneğin *restrict* gösterici bir diziyi gösteriyorsa yalnızca o göstericinin gösterdiği elemana değil, o dizinin hiçbir elemanına başka bir yolla erişmeyeceksiniz.