CS360 Ders notları – Echo

- Jian Huang
- CS360
- Dizin: **~huangj/cs360/notes**
- Ders notları: **http://www.cs.utk.edu/~huangj/cs360/360/notes/Echo/echo.html**

Gerçek İş/Uğraş

Size tüm gerekli ısınma egzersizleri verildiğine göre, profesyoneller tarafından yapılmış bir gerçek dünya programına göz atmaya hazırsınız. Öğrenilecek pek çok şey var. Bu nedenle dikkat edin. Örneğimiz, kod uygulama yankısı (bakınız, *man echo*).

```
/*
 * Telif Hakkı (c) 1989, 1993
 *    California Üniversitesi Yönetim Kurulu. Tüm hakları saklıdır.
 *
 * Kaynak ve ikili formların, değişiklik yapılarak veya yapılmayarak,
 * yeniden dağıtım ve kullanımlarına aşağıda sıralanan şartlarla izin
 * verilmiştir:
 * 1. Kaynak kodun yeniden dağıtımları yukarıdaki telif hakları bildirimini, bu
 *    şartlar listesini ve aşağıdaki feragatnameyi korumalıdır.
 * 2. İkili formun yeniden dağıtımları, yukarıdaki telif hakları bildiriminin,
 *    bu şartlar listesinin ve dokümantasyonda ve/veya dağıtımla sağlamış diğer
 *    materyallerde bulunan aşağıdaki feragatnamenin kopyasını barındırmalıdır.
 * 3. Yayınlar veya bu yazılımın kullanımından bahseden tüm tanıtım materyalleri
 *    aşağıdaki alıntıyı göstermelidir:
 *      Bu ürün, California Üniversitesi-Berkeley- ve onun yazarları
 *      tarafından geliştirilmiş yazılım içermektedir.
 * 4. Ne Üniversite'nin ne de onun yazarlarının adı, bu yazılımdan türetilen
 *    ürünleri onaylamada veya desteklemede, kendine özgü önceden yazılı izin
 *    olmadan kullanılamaz.
 *
 * BU YAZILIM, YÖNETİM KURULU ÜYELERİ VE YAZARLARI TARAFINDAN "ŞU ANKİ HALİYLE"
 * SAĞLANMIŞTIR VE -AMA SINIRSIZ…- İBARESİ İÇEREN HİÇBİR AÇIK VEYA KASDEDİLMİŞ
 * GARANTİ, TİCARETE VE BELİRLİ BİR AMACA UYGUNLUĞU KASTEDİLMİŞ GARANTİLER
 * ONAYLANMAZ. BU YAZILIMIN KULLANIMINDAN DOLAYI HERHANGİ BİR YOLLA ORTAYA
 * ÇIKAN, DOĞRUDAN, DOLAYLI, TESADÜFÎ, ÖZEL, ÖRNEK NİTELİĞİNDE VEYA SONUCA BAĞLI
 * HERHANGİ ZARAR (AMA SINIRSIZ… , ÜRÜN DEĞİŞİMİ VEYA SERVİS TEDARİKİ; İŞ, VERİ
 * VEYA KÂR KAYBI VEYA İŞ KESİNTİSİ, DURUMLARINI İÇEREN)OLAN HİÇBİR OLAYDA HER
 * NE ŞEKİLDE SEBEP OLURSA OLSUN, HERHANGİ BİR SORUMLULUK TEORİSİ ÜZERİNDEN
 * -SÖZLEŞMEDE OLMAYAN, TAM SORUMLULUK VEYA HAKSIZ MUAMELE (İHMAL VEYA DİĞER
 * DURUMLAR DÂHİL)- BÖYLE HASAR OLASILIĞI BİLDİRİLMİŞ BİLE OLSA, YÖNETİM KURULU
 * VEYA YAZARLAR SORUMLU OLMAYACAKTIR.
 */

#include < sys/cdefs.h >
#ifndef lint
__COPYRIGHT(
"@(#) Copyright (c) 1989, 1993\n\
        The Regents of the University of California.  All rights reserved.\n");
#endif /* not lint */

#ifndef lint
#if 0
static char sccsid[] = "@(#)echo.c    8.1 (Berkeley) 5/31/93";
#else
__RCSID("$NetBSD: echo.c,v 1.7 1997/07/20 06:07:03 thorpej Exp $");
#endif
```

```
#endif /* not lint */

#include < stdio.h >
#include < stdlib.h >
#include < string.h >

int     main __P((int, char *[]));

int
main(argc, argv)
        int argc;
        char *argv[];
{
        int nflag;

        /* This utility may NOT do getopt(3) option parsing. */
        if (*++argv && !strcmp(*argv, "-n")) {
                ++argv;
                nflag = 1;
        }
        else
                nflag = 0;

        while (*argv) {
                (void)printf("%s", *argv);
                if (*++argv)
                        putchar(' ');
        }
        if (!nflag)
                putchar('\n');

        exit(0);
}
```

Dikkat, *main* ile ilgili bir püf nokta kullanılıyor. Bu, her ne kadar eski bir derleyici kullanılsa da - ANSI C'1987' den eski tarihliler dahil-, bu kodun derlenip çalışacağından emin olmak içindir. Bu sorun tipik olarak uygulama programcıları tarafından dikkate alınmaz. Burada, *__P* , NULL döndürmek veya hiçbir şey yapmamak için tanımlanmış basit bir makrodur. Örneğin şunun gibi:

```
#define __P(s) ()
#define __P(s) s
```

Elbette, geliştiriciler daima okunaklı kod hedeflemelidirler. *__P* , aldatıcı bir fonksiyon prototipi anlamına geldiğine göre, *__P* 'ler aşağıdaki gibi de tanımlanabilir:

```
#define __P(protos) ()
#define __P(protos) protos
```

Sonraki adım kod gövdesi üzerinde durmak ve bu özlü kod parçasının nasıl şık bir şekilde görevi ele aldığını görmektir.

```
{
        int nflag;

        /* This utility may NOT do getopt(3) option parsing. */
        if (*++argv && !strcmp(*argv, "-n")) {
                ++argv;
                nflag = 1;
        }
        else
```

```
                nflag = 0;

        while (*argv) {
                (void)printf("%s", *argv);
                if (*++argv)
                        putchar(' ');
        }
        if (!nflag)
                putchar('\n');

        exit(0);
}
```

Hata denetiminin nasıl yapıldığına dikkat edin. Echo, içeriğini karşılaştırma için kullanmadan önce, ilk olarak bir *argv argümanının varlığını doğrular. Biliyoruz ki dönüş değerleri çok önemli bir bilgi kaynağıdır. Genellikle böyle bilgileri kullanmaya ihtiyacınız vardır. Fakat basit bir şekilde onu kaybetmediğinizden emin olmak için, tıpkı dönüş değerinin void atanması gibi, bu gibi kasıtlı ihmalleri açık/belirgin yapın.

```
                (void)printf("%s", *argv);
```

Buna rağmen *printf* gerçekten 0 döndürürse, yazılacak bir hata anlamına gelir mi? Veya *putchar* benzer sebeple EOF döndürse? Bu senaryo geçekleşebilir, örneğin çıktı bir dosyaya yeniden gönderildiğinde ve disk alanını henüz tüketmişseniz... Bu durumda ne olur?

İşlevsel Olmayan İçerikler

Lisansı kapsayan kodun üst kısmı, serbest bırakılmış kodun altındadır. Bu, günümüz açık kaynak geliştirme genel hareketine verilen önemli bir yayındır. Ancak biz bu tartışmayı dönem sonuna saklayacağız.

Acemileri şaşırtan/yanıltan lisans açıklamasının tam altındaki kodun birkaç satırıdır.

```
#include < sys/cdefs.h >
#ifndef lint
__COPYRIGHT(
"@(#) Copyright (c) 1989, 1993\n\
        The Regents of the University of California.  All rights reserved.\n");
#endif /* not lint */

#ifndef lint
#if 0
static char sccsid[] = "@(#)echo.c    8.1 (Berkeley) 5/31/93";
#else
__RCSID("$NetBSD: echo.c,v 1.7 1997/07/20 06:07:03 thorpej Exp $");
#endif
#endif /* not lint */
```

Öncelikle biraz basitleştirelim. *lint*, kodda potansiyel problemleri, uyarı noktalarını kontrol eden bir araçtır. Mutlak programlama davranışına benzemez. Ancak, echo karmaşık olmadığından, çok kritik değildir. Yani, birkaç başka parça gibi *lint* ile ilgili parçalar da çıkarıldığında, şunu elde ederiz:

```
#include < sys/cdefs.h >
__COPYRIGHT(
"@(#) Copyright (c) 1989, 1993\n\
```

```
        The Regents of the University of California.  All rights reserved.\n")

__RCSID("$NetBSD: echo.c,v 1.7 1997/07/20 06:07:03 thorpej Exp $");
```

Daha iyi bakın. Şimdi, < sys/cdefs.h > kısmına bakın, biliyorsunuz ki *__COPYRIGHT()* ve *__RCSID()* sadece bir string' i global bir değişken olarak bildiren bazı makrolardır. Basit yeterlidir. Fakat niye?

Burada yine sistem programcılarının bakış açısı var. Mekanikçilik, davranış biçimi değildir. Gerçekten de diğer insanlar kodunuzu kullanacak. Sahip olduklarının tümü çalıştırılabilir şeyler olursa, hangi versiyonu kullanıyorlar? Telif hakkı kimde? Vs. nasıl bilebilirler?

"Sadece çalıştır ve "Hakkında" düğmesini tıkla" diyorsunuz. :) Fakat eğer çalıştırılabilir olan, yeni bir bellek kartının bir sistem çağrısı tarafından yardım alınması gereken aygıt sürücüsü olarak kullanılıyorsa? Bu karışıklık aralıksız olarak yayılabilir. İlkenin/hareket tarzının ne olduğunu tanımlamak istemiyoruz. Sonra sadece bu bilgileri object dosyalar ve çalıştırılabilir dosyalara yerleştirmek son derece kullanışlı görünür.

Sonra, onlara *strings* ve *ident*, vb. gibi basit komutları kullanırken bakabilirsiniz.

Sonraki satır nedir?

```
"$NetBSD: echo.c,v 1.7 1997/07/20 06:07:03 thorpej Exp $"
```

Yazdığınız şeylere benzemiyor. Gerçi ben, sadece mükemmel görünmek için buna benzer satırları kodlarının içine yazan öğrenciler gördüm. Genç arkadaşların pek çok boş zamanı var gibi görünüyor.

Bunlar versiyon kontrol araçlarındandır -özellikle RCS (revizyon kontrol sistemi)-. Tüm yazılım uzmanları versiyon kontrol araçlarının güvenilir bir çeşidini kullanır. RCS, hepsinin temelidir. Temelleri gerçekten tamamıyla aynı olsa da sonraki araçlar daha çok fonksiyonlu ve daha gelişmiş. RCS öğrenmek iyi bir başlangıçtır. Lab3 ödevlerinden başlayarak, tüm kodlarınız RCS içinde korunmalıdır. Makefile' ınızı da, bir seçenekle kodunuzu RCS' den ayırabilecek şekilde yapmalısınız. Bu şekilde çalışmalıdır:

```
UNIX>make checkout
UNIX>make all
```

Daha fazla bilgi için, *man rcsintro* , *man ci* ve *man co' yu* okuyun.

Son olarak, Unix programlamayı öğrenmemize rağmen, taşınabilirlik hâlâ dikkat etmemiz gereken önemli bir olaydır. Her ne zaman < sys/... > altındaki header dosyalarını yüklerseniz sadece Unix' e uygun anlamına gelir. Echo' nun daha taşınabilir bir versiyonu burada sağlanmaktadır, onun gibi oldukça modern bir versiyonu da burada bulabilirsiniz.