

GCC Kullanımı

GCC harfleri **GNU Compiler Collection** kelimelerinin baş harflerinden oluşur. Gerçek ismini **GNU C Compiler** kelimelerinden almaktadır. Bu değişikliğin sebebi GCC eskiden sadece C derleyicisinden oluşmakta olduğudur. Diğer diller daha sonradan GCC ailesine eklenmiştir. Bu konuda bilgi için ;

http://en.wikipedia.org/wiki/GNU_Compiler_Collection adresine bir ziyaret yapılabilir.

<http://gcc.gnu.org/> adresinden GCC son sürümleri takip edebilirsiniz.

GCC sürümünüzü öğrenmek için;

```
bash-2.05b# gcc -v
```

```
Reading specs from /usr/lib/gcc-lib/i386-slackware-linux/3.2.2/specs
```

```
Configured with: ../gcc-3.2.2/configure --prefix=/usr --enable-shared  
--enable-threads=posix --enable-__cxa_atexit --disable-checking --with-  
gnu-ld --verbose --target=i386-slackware-linux --host=i386-slackware-  
linux
```

```
Thread model: posix
```

```
gcc version 3.2.2
```

Benim sistemimde dönen sonuç yukarıdaki şekildedir.

Derleyicinin uygulamaları doğru bir şekilde derlemesi için bir takım parametrelerin doğru olarak verilmesi gerekmektedir. En basit şekli ile uygulamamızı yazalım ve ilk derleme işlemimize geçelim.

```
#include <stdio.h>  
int main()  
{  
printf("Örnek uygulama");  
}
```

Yukarıdaki en basit uygulamamızdan başlamış olduk. Şimdi bu uygulamamızı derlemeye geçelim. Uygulamayı istediğimiz bir editör ile yazdıktan sonra `ilk.c` olarak kaydedelim ve aşağıdaki satırları yazalım.

```
#gcc ilk.c
```

Bu satırdan sonra uygulamada hata yok ise veya bir uyarı mesajı vermezse uygulamalar derlenir. Kaynak kodun olduğu dizine bakıldığı zaman **a.out** isimli bir dosyanın oluştuğu görülecektir. Uygulamamızı bir isim vermeden derledik bu gibi durumlarda **gcc** varsayılan olarak `a.out` dosya adını kullanacaktır. Şimdide uygulamamızı kendi ismiyle derleyelim. Bu amaçla `-o` parametresi kullanılacaktır. `-o` parametresi çıkış dosya adını belirtmektedir.

```
#gcc ilk.c -o ilk
```

Komut satırından sonra ilgili dizin içerisinde artık bir `ilk` isimli çalıştırılabilir dosya oluşacaktır. Bu aşamadan sonra artık basit bir uygulamanın derlenmesi işlemi anlaşılabilir oldu. Peki ya uygulama içerisinde dışarıdan bir takım kitaplıklar eklenmek istenirse nasıl

derlenmelidir? Bu aşamada örnek basit uygulamamız üzerinde bir iki ufak değişiklik yapalım;

```
#include <ncurses.h>
int main()
{
    initscr();
    getch();
    endwin();
    printf("bitti\n");
}
```

Standart kitaplıklar kullanıldığı zaman `getch()` fonksiyonu yoktur. Ancak eğer `ncurses` kullanılması durumunda bu komut mevcuttur. Dikkat edilecek olursa `stdio.h` uygulamaya eklenmedi. Ön bilgi olarak `ncurses.h` kullanılması durumunda `stdio.h` kullanılmasına gerek yoktur. Bu uygulamayı aynı şekilde aşağıdaki ilk öğrendiğimiz standart derleme yöntemi ile derleyelim.

```
# gcc ilk.c -o ilk
/tmp/cchah5Wx.o: In function `main':
/tmp/cchah5Wx.o(.text+0x7): undefined reference to `initscr'
/tmp/cchah5Wx.o(.text+0x10): undefined reference to `stdscr'
/tmp/cchah5Wx.o(.text+0x15): undefined reference to `wgetch'
/tmp/cchah5Wx.o(.text+0x1d): undefined reference to `endwin'
collect2: ld returned 1 exit status
```

GCC bize bir çok tanımlanmamış komut olduğunu hata olarak vermektedir. Aslında bunlar hata değil bunu biliyoruz ancak bu komutlar `ncurses` kitaplığı içerisinde. Bizim yapmamız gereken `ncurses` kitaplığının bayrağını derleme parametresine eklemek olacaktır.

```
# gcc ilk.c -o ilk -lncurses
```

Bu komuttan sonra uygulama hata vermeden derleme işlemini gerçekleştirecektir. Önemli hatırlatma, uygulamaya eklenecek olan kitaplıklar ve kütüphaneler eğer sistemde standart olan yerlerde mevcut ise ki bu standart yerler nerelerdir. Bu tanım dağıtımlara göre farklılıklar gösterebilir ancak tüm kütüphanelere ait ekleme dizinleri genel olarak `/usr/include` içindedir. Eğer bu başlık dosyaları farklı bir yerde oldukları belirtilmek istenmesi durumunda farklı bir parametre daha eklemektedir.

```
# gcc ilk.c -o ilk -I/usr/include -lncurses
```

satırı da aynı şekilde uygulamamızı sorunsuz olarak derleyecektir. Burada uygulamamız içine eklenen başlık dosyalarını araması için standart path dışında `/usr/include` dizinini de eklemiş olduk. Şu aşamadan itibaren `ncurses` kitaplıklarını uygulamamız içerisine ekleyebiliyoruz ve istediğimiz `ncurses` komutlarını kullanabiliriz. Bunun yanında `mysql` kitaplıklarını da uygulamaya katmak istersek uygulamamızı aşağıdaki şekle sokalım;

```
#include <ncurses.h>
#include <mysql.h>
int main()
{
    initscr();
    getch();
    endwin();
    printf("bitti\n");
}
```

En son derleme satırımız ile derleyelim. Aynı zamanda `-lmysql` ile mysql kitaplıklarını kullanacağımızı belirtelim.

```
# gcc ilk.c -o ilk -I/usr/include -lncurses -lmysql
ve sonuç;
```

```
ilk.c:2:19: mysql.h: No such file or directory
neden mysql.h bulamadı halbuki -I/usr/include yazdık, peki sistemde ufak bir arama yapalım bakalım doğrumu yazdık?
```

```
# locate mysql.h
/usr/share/apps/quanta/doc/php/ref.mysql.html
/usr/include/mysql/mysql.h
```

işte aradığımız satır. `mysql.h` dosyası `/usr/include/mysql` dizini için de bulunmakta imiş öyleyse derleme parametresinde bir değişiklik yapalım.

```
# gcc ilk.c -o ilk -I/usr/include -I/usr/include/mysql -lncurses -lmysql
```

ve sonuç başarılı olacaktır. Bu şekilde istendiği kadar kitaplığı uygulamamız içerisine ekleyebiliriz. Eğer aranan uygulama farklı bir dizin içerisinde ise öyleyse `-L` parametresi ile bu kütüphanenin yolunu tanımlamalıyız. Örnek olarak;

```
# gcc ilk.c -o ilk -I/usr/include -I/usr/include/mysql -L/usr/local/lib
-lncurses -lmysql
```

Aynı şekilde matematik kütüphanesini uygulamamızda kullanacaksak `-lm` eklemeliyiz. şifreleme algoritmasını kullanacaksak `-lcrypt`, postgresql için `-lpq`, thread kullanımı için `-lpthread`, glib kütüphanesini kullanmak için `-lglib`, vga kitaplıklarını kullanmak için `-lvga` örnek olarak verebiliriz. Bu kullanım örnekleri sisteminize kurduğunuz kitaplıkların miktarına göre değişmektedir. Şimdide GCC ile kullanabileceğimiz bir takım parametrelere gelelim. `-m` parametresi CPU seçimli olara derleme işlemini gerçekleştirmektedir.

386 komut seti için **-m386**

486 komut seti için **-m486**

pentium komut seti için `-mpentium` parametresi verilmelidir. Örnek olarak;

```
# gcc ilk.c -o ilk -I/usr/include -I/usr/include/mysql -lncurses
-lmysql -mpentium
```

komutu sonrası uygulama pentium işlemci için derlenmiş olacaktır. 486 seçimli derleme işleminden sonra özellikle büyük uygulamalardan sonra kodda bir miktar büyüme olabilir, ancak bu hız ile ters orantılı olarak artırmaktadır. Diğer önemli bir kullanım şekli uygulamaya eklenmiş olan kitaplıkları statik olarak uygulama içerisine eklemektir. Bu şekilde uygulama çalıştırılabilir kod içerisine kitaplıklarda eklenecektir. Bu işlemi örnek uygulamamız içinde deneyelim;

```
# gcc ilk.c -o ilk -I/usr/include -I/usr/include/mysql -lcurses  
-lmysql -mpentium -static
```

Her iki farklı derleme şekline sonra uygulamanın büyüklüğüne dikkat ediniz, büyüklüğünde epey bir fark olduğu görülecektir. Uygulama içerisinde verilen hata mesajları haricinde bir takım uyarı mesajları bulunmaktadır. Örnek olarak göstereceğimiz var olan değişkenin tanımlanmış olmasına rağmen kullanılmaması gibi; Uygulama içerisinde bir değişken tanımlayalım ve derleyelim; Derleme parametresine ek olarak -Wall parametresini ekleyelim.

```
gcc ilk.c -o ilk -I/usr/include -I/usr/include/mysql -lcurses -Wall  
ilk.c: In function `main':  
ilk.c:5: warning: unused variable `a'  
ilk.c:10: warning: control reaches end of non-void function
```

aşağıdaki şekilde karşımıza uyarı mesajları çıkmaya başlayacaktır. Eğer uygulama içerisinde optimizasyon yapılmasına önem verilmesi durumunda bu gereksiz olan değişkenler temizlenmelidir. -w parametresi ile birlikte tüm uyarı mesajlarının ekrana çıkmasına engel olunur. GCC aynı zamanda kendi içerisinde yazılmış olan kodlar üzerinde optimizasyon yapmaktadır. Bu amaçla -O parametresi kullanılmaktadır. Örnek uygulamamızı aşağıdaki şekilde derleyelim.

```
# gcc ilk.c -o ilk -I/usr/include -I/usr/include/mysql -lcurses -O1
```

Bu parametre -O0, -O1, -O2, -O3 değerlerini alabilir. 0 hiç optimize yaptırmazken 3 en optimize hali ile uygulama kodunu düzenler.

M.Ali VARDAR

ali@linuxprogramlama.com

Bu yazının son şeklini www.linuxprogramlama.com adresinden temin edebilirsiniz.

Yasal Açıklama:

Bu belgenin, [GCC Kullanımı]1.0 sürümünün telif hakkı © 2005 M. Ali Vardar'a aittir. Bu belgeyi, Free Software Foundation tarafından yayınlanmış bulunan GNU Özgür Belgeleme Lisansı1.1 ya da daha sonraki sürümünün koşullarına bağlı olarak kopyalayabilir, dağıtabilir ve/veya değiştirebilirsiniz. Bu Lisansın bir kopyasını <http://www.gnu.org/copyleft/fdl.html> adresinde bulabilirsiniz.

BU BELGE "ÜCRETSİZ" OLARAK RUHSATLANDIĞI İÇİN, İÇERDİĞİ BİLGİLER İÇİN İLGİLİ KANUNLARIN İZİN VERDİĞİ ÖLÇÜDE HERHANGİ BİR GARANTİ VERİLMEMEKTEDİR. AKSİ YAZILI OLARAK BELİRTİLMEDİĞİ MÜDDETÇE TELİF HAKKI SAHİPLERİ VE/VEYA BAŞKA ŞAHISLAR BELGEYİ "OLDUĞU GİBİ", AŞIKAR VEYA ZİMNEN, SATILABİLİRLİĞİ VEYA HERHANGİ BİR AMACA UYGUNLUĞU DA

DAHİL OLMAK ÜZERE HİÇBİR GARANTİ VERMEKSİZİNDAĞITMAKTADIRLAR. BİLGİNİN KALİTESİ İLE İLGİLİ TUM SORUNLAR SİZE AİTTİR. HERHANGİ BİR HATALI BİLGİDEN DOLAYI DOĞABİLECEK OLAN BÜTÜN SERVİS, TAMİR VEYA DÜZELTME MASRAFLARI SİZE AİTTİR. İLGİLİ KANUNUN İCBAR ETTİĞİ DURUMLAR VEYA YAZILI ANLAŞMA HARİCİNDE HERHANGİ BİR ŞEKİLDE TELİF HAKKI SAHİBİ VEYA YUKARIDA İZİN VERİLDİĞİ ŞEKİLDE BELGEYİ DEĞİŞTİREN VEYA YENİDEN DAĞITAN HERHANGİ BİR KİŞİ, BİLGİNİN KULLANIMI VEYA KULLANILAMAMASI (VEYA VERİ KAYBI OLUŞMASI, VERİNİN YANLIŞ HALE GELMESİ, SİZİN VEYA ÜÇÜNCÜ ŞAHISLARIN ZARARA UĞRAMASI VEYA BİLGİLERİN BAŞKA BİLGİLERLE UYUMSUZ OLMASI) YÜZÜNDEN OLUŞAN GENEL, ÖZEL, DOĞRUDAN YA DA DOLAYLI HERHANGİ BİR ZARARDAN, BÖYLE BİR TAZMİNAT TALEBİ TELİF HAKKI SAHİBİ VEYA İLGİLİ KİŞİYE BİLDİRİLMİŞ OLSA DAHİ, SORUMLU DEĞİLDİR.

Tüm telif hakları aksi özellikle belirtilmediği sürece sahibine aittir. Belge içinde geçen herhangi bir terim, bir ticari isim ya da kuruma itibar kazandırma olarak algılanmamalıdır. Bir ürün ya da markanın kullanılması ona onay verildiği anlamında görülmemelidir.