GNU Debugger Kullanımı

Yazan:
M. Ali Vardar
<ali (at) linuxprogramlama.com>

Ocak 2005

Özet

Uygulamalarımızı geliştirme sırasında gerek sistemden olsun gerek yazılımcı tarafından olsun gelen bir takım sinyaller veya kesmeler veya hatalar yüzünden uygulamamızın çalışmasında kesilmeler olabilir. Bu gibi durumları çoğu zaman tahmin edebiliyor olmak yeterli olmayabilir. Bu gibi durumlarda en büyük yardımcımız gdb olacaktır.

Geçmiş

1.0	Ocak 2005	MAV

Belgenin özgün sürümü `http://www.linuxprogramlama.com` adresinde bulunabilir.

Yasal Açıklamalar

GDB Kullanımı

Uygulamalarımızı geliştirme sırasında gerek sistemden olsun gerek yazılımcı tarafından olsun gelen bir takım sinyaller veya kesmeler veya hatalar yüzünden uygulamamızın çalışmasında kesilmeler olabilir. Bu gibi durumları çoğu zaman tahmin edebiliyor olmak yeterli olmayabilir. Bu gibi durumlarda en büyük yardımcımız **gdb** olacaktır. Linux sistemlerde yaygın olarak GBU debugger isimli uygulama kullanılır. Bu uygulama ile uygulamanızın kodu veya `core` dosyası incelenebilir. Öncelikle derleme işlemine bir bakalım.

```
$ gcc -g deneme.c -o deneme
```

Derleme işleminden sonra uygulamanın büyüklüğünün normal `-g` parametresiz büyüklüğünden fazla olduğu görülecektir. Bu aşamadan sonra derlenen uygulama hata ayıklama işlemleri için gerekli olan açıklamaları da çalıştırılabilir dosyanın içerisine katmaktadır. Hata veren örnek bir uygulama yazalım. Örnek uygulama içerisinde değişiklik yapalım ve uygulamanın hata vermesini sağlayalım.

```
int main()
{
  int a[5];
  int i=0;
  duzenle(a);
}

void duzenle(int a[])
{
  int i;
  for (i=0; i<10000; i++) a[i]=i;
}
```

Uygulamamız `for (i=0; i<10000; i++) a[i]=i;` satırında hata vermektedir. Çünkü bu satırda hafıza için ayrılmış olan 5 tamsayılık alanın 10000 nci değerine erişilmeye çalışılmaktadır. Uygulama ilk çalıştırıldığı anda "segmentation failed" (bölütleme başarısız) hatası verecektir. Bu kadar kısa bir kod içerisinde bunu bulmak sorun olmayacaktır. Ancak uygulamanın iç içe çağrılardan oluşan 100.000 satırlık bir uygulama olduğunu varsayarsanız bu gibi bir hatanın bulunması yazılımcıyı epey zorlayacaktır. Uygulamaya tekrar geri dönelim ve uygulamayı şimdi aşağıdaki şekilde çalıştıralım.

```
[root@linuxprogramlama 1]# gdb deneme
GNU gdb Red Hat Linux (5.1.90CVS-5)
Copyright 2002 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and
you are welcome to change it and/or distribute copies of it under
certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.     Type "show warranty" for
details.
This GDB was configured as "i386-redhat-linux"...
(gdb)
```

gdb ilk çalıştığı zaman yukarıdaki şekilde bir karşılama iletisi karşımıza çıkacaktır. Uygulama çalıştırılabilir dosyası **gdb** tarafından yüklenmiş olacaktır. Uygulamayı **r** komutunu verelim ve karşımıza çıkan iletiye bir bakalım.

```
(gdb) r
Starting program: /Projects/Kurs/1/deneme
Program received signal SIGSEGV, Segmentation fault.
0x08048418 in duzenle (a=0xbffff970) at deneme.c:12
12 for (i=0; i<9000; i++) a[i]=i;
```

İşte **gdb** hata veren satırımızı ekrana yazdı. Açıklamanın başında bulunan 12 rakamı bize uygulama kodumuz içerisinde hatayı veren satırdır. Şimdi hatanın oluşma süreçlerine bakalım bu amaçla **bt** (backtrace) komutunu kullanıyoruz.

```
(gdb) bt
#0  0x08048418 in duzenle (a=0xbffff970) at deneme.c:12
#1  0x080483e9 in main () at deneme.c:6
#2  0x0000000b in ?? ()
Cannot access memory at address 0xa
(gdb)
```

Böylelikle en sade halde uygulamamızda bulunan hataları izleme yöntemini gördük. Şimdi de daha etkin bir şekilde GNU debugger kullanabilmek amacıyla bir takım komutlara göz atalım.

list komutu bize hataya neden olan satırın bulunduğu komut listesini verecektir.

```
(gdb) list
7       }
8
9       void duzenle(int a[])
10      {
11      int i;
12      for (i=0; i<9000; i++) a[i]=i;
13      }
(gdb)
```

Değişkenler hakkında bilgi almak için **print** komutu kullanılır.

```
(gdb) print a[5]
$2 = 5
(gdb)
```

Değişken türünü öğrenmek için **whatis** komutu kullanılır.

```
(gdb) whatis a
type = int *
```

Peki bir koşul oluşması durumunda istenen değişken bilgilerini almak istesek ne yapmamız gerekir? Bu amaçla uygulamanın çalışma anında durdurulmasını sağlayacak bir kırılma noktası yerleştirmeliyiz.

```
(gdb) break 12
Breakpoint 1 at 0x80483f6: file deneme.c, line 12.
(gdb) r
Starting program: /Projects/Kurs/1/deneme

Breakpoint 1, duzenle (a=0xbffff970) at deneme.c:12
12 for (i=0; i<9000; i++) a[i]=i;
(gdb) print i
$1 = 0
```

Yukarıda, 12. satıra bir kırılma noktası yerleştirdik ve o anki i değerine baktık. Diğer bir şartlı kırılma noktasına örnek ise;

```
(gdb) break 12 if i==5
Breakpoint 1 at 0x80483f6: file deneme.c, line 12.
(gdb)
```

Görüldüğü üzere *i* değişkeni 12. satırda 5 değerine ulaştığı an uygulama kesilecektir. Makina dili kodlarını öğrenmek amacıyla verilen iki bellek arasını yazdırmak istersek:

```
(gdb) disas 0x63e4 0x6404
Dump of assembler code from 0x63e4 to 0x6404:
0x63e4 <builtin_init+5340>:       ble 0x63f8 <builtin_init+5360>
0x63e8 <builtin_init+5344>:       sethi %hi(0x4c00), %o0
0x63ec <builtin_init+5348>:       ld [%i1+4], %o0
0x63f0 <builtin_init+5352>:       b 0x63fc <builtin_init+5364>
0x63f4 <builtin_init+5356>:       ld [%o0+4], %o0
0x63f8 <builtin_init+5360>:       or %o0, 0x1a4, %o0
0x63fc <builtin_init+5364>:       call 0x9288 <path_search>
0x6400 <builtin_init+5368>:       nop
End of assembler dump.
```

Kesilmiş olan yerden devam etmek aşağıdaki gibi **c** komutunu vermek gerekir.

```
(gdb) c
Continuing.
```

gdb'den çıkmak için **quit** veya **q** yazınız.

Aynı zamanda uygulamanız çöktüğünde oluşan `core` dosyasını da

```
# gdb core
```

şeklinde kullanarak hata veren yeren yeri bulabilirsiniz.

Notlar

a) Belge içinde dipnotlar ve dış bağlantılar varsa, bunlarla ilgili bilgiler bulundukları sayfanın sonunda dipnot olarak verilmeyip, hepsi toplu olarak burada listelenmiş olacaktır.

b) Konsol görüntüsünü temsil eden sarı zeminli alanlarda metin genişliğine sığmayan satırların sığmayan kısmı ¬ karakteri kullanılarak bir alt satıra indirilmiştir. Sarı zeminli alanlarda ¬ karakteri ile başlayan satırlar bir önceki satırın devamı olarak ele alınmalıdır.

Bu dosya (gdb–NASIL.pdf), belgenin XML biçiminin TEXLive ve belgeler-xsl paketlerindeki araçlar kullanılarak PDF biçimine dönüştürülmesiyle elde edilmiştir.

6 Şubat 2007