```
# gdb için kullanılacak çalışabilir programın -g switch ile derlenmesi gerekir

#Çalışabilir dosyanın sembollerini yükler.

gdb sample

#gdb içerisinde daha sonra başka bir dosya yüklemek için
file mample

#gdb içerisinde bash komutu çalıştırmak için shell kendi komutu kullanılır

shell clear
shell gcc -g -o sample sample.c

#Programın komut satırı argümanları set args komutu ile verilir
set args ali veli selami

#r veya run ile program çalştırılabilir. run programda breakpoint görmezse durmaz.

r

#start ile program çalıştırılır. main fonksiyonuna geçici breakpoint konulmuş kabul edilir. Yani main de akış durur

#c veya continue ile break point de duran program devam ettirilebilir

#Kalıcı breakpoint için b veya break kullanılır

#Geçici breakpoint için tb veya tbreak kullanılır

#breakpoint şu şekillerde koyulabilir:

#b nin doğrudan kullanımı program çalışıyor ise bir sonraki adıma koyar

#b satır numarası ile ilgili satır numarasına konabilir

#include <stdio.h>

int main(int argc, char **argv)
{
    int i;

    for (i = 0; i < argc; ++i)
        puts(argv[i]);

    printf("End of main\n");

    return 0;
}

#b fonksiyon ismi: O dosyadaki o fonksiyona breakpoint koyar. Şüphesiz C++ da overloading olabilir. Dikkat edilmesi gerekir. Bu durumda overloading için parametrik yapı da verilmelidir. İleride ele alınacaktır

#b sample.c:foo durumunda ilgili modüldeki foo ya konur
```

```
53
54  #b sample.c:10 sample.c deki satıra konur
55
56  #b *0x12345: adrese break point konur
57
58  #Tüm breakpoint ler delete komutu ile silinebilir
59
60  #Bir processe signal göndermek için signal komutu kullanılır
61
62  #call komutu ile process içerisinde çağrılabilen (genelde sistem
      fonksiyonları) fonksiyonlar çağrılabilir.
63
64  #breakpoint ler hakkında bilgi almak için info breakpoint komutu kullanılır
65
66  #kill komutu ile process sonlandırılabilir
67
68  #step (s) komutu ile tek bir satır çalıştırılabilir. Fonksiyon içine girer
      (step into)
69
70  #next (n) komutu yine tek bir çalıştırır. Fonksiyon içine girmez
71
72  #watch komutu ile herhangi bir nesne izlenebilir.
73
74  #C++ da bir fonksiyona breakpoint koymak için fully qualified ismi
      verilmelidir. Örneğin: b csd::Sample::foo(double)
75
76  #b ile koşullu break yapılabilir. Aynı durum watch için de geçerlidir
77
78  #watch işleminde iki şekil kullanılabilir.
79
80  #watch val
81
82  #watch *val
83
84  #diziler de watch işlemiyle izlenebilir
85
86  #watch a[i]
87
88  Örneğin: watch x if x>0
89
90  #içiçe aynı isimli değişkenlerde kapsayan bloktaki ismin takibi * ile
      yapılması önerilir.
91  Örneğin:
92
93  #include <iostream>
94
95  using namespace std;
96
97  int val;
98
99  int main(int argc, char **argv)
100 {
101     int val;
102     int *p = &val; // Val içteki blokta izlenecekse * izlemesi yapılmalı
103
104     {
```

```
105       int val;
106
107       val = 40;
108
109       *p = 30;
110
111       cout << val << endl; //
112     }
113
114     cout << val << endl;
115
116
117     return 0;
118 }
119
120 #analiz ediniz:
121
122 #include <iostream>
123 #include <cstring>
124 #include <cstdlib>
125 #include <ctime>
126
127 using namespace std;
128
129 const char g_text[] = "0123456789abcdef";
130 const int g_len = 13;
131
132 void overrun(char *buf, const char *msg, int len)
133 {
134   char dummy[4096];
135
136   memset(dummy, len, sizeof(dummy));
137
138   memcpy(buf, msg, len);
139 }
140
141 int main(int argc, char **argv)
142 {
143     char *buf = new char[g_len];
144     int i;
145
146     srand(static_cast<unsigned int>(time(0)));
147
148     int n = 1000;
149
150     int thresh = RAND_MAX / n;
151
152     for (i = 0; i < n; ++i) {
153       int len = rand() < thresh ? g_len + 1: g_len;
154
155       overrun(buf, g_text, len);
156     }
157
158     int overran = strlen(buf) > g_len;
159
160     if (overran)
```

```
161       cout << "overrun" << endl;
162     else
163       cout << "No overrun" << endl;
164
165     delete buf;
166
167
168     return 0;
169 }
170
171
172 #bt veya backtrace komutu ile çağrıma stack inin komutuna bakılır.
173
174 #info locals komutu o an (context) bulunan yerel değişkenleri gösterir
175 #whatis komutu bir sembolün hakkında bilgi verir
176
177 #printf(p) komutu C de printf gibi düşünülebilir. argüman olarak aldığı
      nesnenin bilgisini terminale verir
178
179 Örneğin: printf "x=%d, y=%lf", x, y
180
181 #examine komutu bir adres için kullanılabilir
182
183 #x için bir takım özel durumlar vardır. Dökümanlar incelenebilir
184
185 #template fonksiyonlar için (template sınıfların fonksiyonu) hangi açılım için
       break point konulacaksa tam ismiyle verilmelidir
186 #include <iostream>
187
188 using namespace std;
189
190 template <typename T>
191 void display(T t)
192 {
193     cout << t << "\n";
194 }
195
196 int main(int argc, char **argv)
197 {
198     int val;
199
200     display(val);
201
202     double d;
203
204     display(d);
205
206     return 0;
207 }
208
209
210
211
212
```