Here we discover that the addresses of global variables are chosen by the compiler, they appear as constants in the executable file.

```
$ cat debugme.cpp

#include <iostream>

int g1, g2;

int oddthing(int a, int b)
{ g1 = a*b;
  g2 = a+b;
  return g1*g2; }

void main()
{ cout << "Address of g1 = " << & g1 << "\n";
  cout << "Address of g2 = " << & g2 << "\n";
  cout << "\n";
  int x = oddthing(12, 34);
  int y = oddthing(123, 345); }


$ CC -ggdb debugme.cpp
$ gdb a.out
(gdb) break oddthing
Breakpoint 1 at 0x804883b: file debugme.cpp, line 6.
(gdb) disas
No frame selected.

(gdb) run
Starting program: /home/students/sratbag/a.out
Address of g1 = 0x8049d38
Address of g2 = 0x8049d3c


Breakpoint 1, oddthing (a=12, b=34) at debugme.cpp:6
6       { g1 = a*b;
(gdb) disas
Dump of assembler code for function _Z8oddthingii:
0x08048838 <_Z8oddthingii+0>:   push   %ebp
0x08048839 <_Z8oddthingii+1>:   mov    %esp,%ebp
0x0804883b <_Z8oddthingii+3>:   mov    0x8(%ebp),%eax
0x0804883e <_Z8oddthingii+6>:   imul   0xc(%ebp),%eax
0x08048842 <_Z8oddthingii+10>:  mov    %eax,0x8049d38
0x08048847 <_Z8oddthingii+15>:  mov    0xc(%ebp),%eax
0x0804884a <_Z8oddthingii+18>:  add    0x8(%ebp),%eax
0x0804884d <_Z8oddthingii+21>:  mov    %eax,0x8049d3c
0x08048852 <_Z8oddthingii+26>:  mov    0x8049d38,%eax
0x08048857 <_Z8oddthingii+31>:  imul   0x8049d3c,%eax
0x0804885e <_Z8oddthingii+38>:  leave
0x0804885f <_Z8oddthingii+39>:  ret
End of assembler dump.
(gdb) quit
```

This is the program that showed us that programs always use the same addresses for their variables, even if the program is being run by two different users at exactly the same time.

```
$ cat sametime.cpp

#include <iostream>

int x;

void main()
{ int y;
  cout << "what should x be ? ";
  cin >> x;
  cout << "what should y be ? ";
  cin >> y;
  string s;
  getline(cin, s);

  int * px = & x;
  int * py = & y;
  int * pz = new int;

  cout << "The address of x is " << (unsigned int)px << "\n";
  cout << "The address of y is " << (unsigned int)py << "\n";
  cout << "              pz is " << (unsigned int)pz << "\n";
  cout << "The address of main is " << (unsigned int)main << "\n\n";

  while (true)
  { cout << "memory location " << (unsigned int)px << " contains " << * px << "\n";
    cout << "memory location " << (unsigned int)py << " contains " << * py << "\n\n";
    cout << "press enter to continue";
    getline(cin, s); } }

$ CC sametime.cpp
$ a.out
what should x be ? 123
what should y be ? 987
The address of x is 134517624
The address of y is 3217026028
              pz is 134533168
The address of main is 134515232

memory location 134517624 contains 123
memory location 3217026028 contains 987

press enter to continue
memory location 134517624 contains 123
memory location 3217026028 contains 987

press enter to continue
```