

### **Primjer 1:**

```
#include<stdio.h>

int findAndReturnMax(int *, int, int);

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

    int arr[5] = { 17, 21, 44, 2, 60};

    int max = arr[0];

    if( findAndReturnMax(arr, 5, max) == 0 ) {
        printf("MAX: %d \n", max);
    }
}

int findAndReturnMax(int *array1, int len, int max){

    int i;

    if(!array1 || (len <= 0)){
        return -1;
    }

    max = array1[0];

    for(i = 0; i < len; i++) {
        if(max < array1[i]){
            max = array1[i];
        }
    }
    return 0;
}
```

-Kompajliramo program za mips arhitekturu:

```
ecc -target mips32r2el-linux -O0 -g max.c -o max
```

-**g** za ecc nalaze kompajleru da u program doda simbole za debugiranje tako da debager moze koristiti imena varijabli i ostalih definicija iz programa.

-Pokrenimo program u simulatoru:

```
root@1c32b766dbcf:/# qemu-mipsel max
```

MAX: 17

Vidimo da je vrijednost varijable max = 17. Međutim najveća vrijednost u nizu arr je 60. Kako bismo otkrili šta nije uredu, koristimo debugger.

-Aktiviramo qemu u posebnom modu tako da čeka na instrukcije debagera na određenom mrežnom portu i proces stavljamo u background:

```
qemu-mipsel -g 1235 max &
```

-g za qemu predstavlja broj mrežnog porta na kojem simulator čeka na konekciju debagera radi kontrole toka izvršenja prosljeđenog programa.

Pokrećemo debager:

```
root@1c32b766dbcf:~# gdb-multiarch max
Reading symbols from max...done.
```

***#--ovdje počinje sesija u koju unosimo sljedeće gdb komande:***

```
(gdb) target remote :1235
Remote debugging using :1235
0x004001a0 in _start ()
```

```
(gdb) break main
Breakpoint 1 at 0x400230: file max.c, line 7. #--postavljamo breakpoint na početak programa
```

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

```
Breakpoint 1, main (argc=1, argv=0x76fff6d4) at max.c:7
7          int arr[5] = { 17, 21, 44, 2, 60};
```

***(gdb) list #--izlistava izvorni kod blizu break pointa***

```
2
3      int findAndReturnMax(int *, int, int);
4
5      int main (int argc, char *argv[]) {
6
7          int arr[5] = { 17, 21, 44, 2, 60};
8
9          int max = arr[0];
10
```

```
11          if( findAndReturnMax(arr, 5, max) == 0 ) {
```

*(gdb) list 11 #--izlistava izvorni kod blizu linije 11*

```
6
7          int arr[5] = { 17, 21, 44, 2, 60};
8
9          int max = arr[0];
10
11          if( findAndReturnMax(arr, 5, max) == 0 ) {
12              printf("MAX: %d \n", max);
13          }
14      }
15
```

*(gdb) next #-- izvrsava sljedecu instrukciju*

```
9          int max = arr[0];
```

*(gdb) #-- enter izvrsava prethodnu komandu*

```
11          if( findAndReturnMax(arr, 5, max) == 0 ) {
```

*(gdb) print max*

```
$1 = 17
```

*(gdb) print max #-- ispisuje vrijednost varijable max*

```
$1 = 17
```

*(gdb) p arr*

```
$2 = {17, 21, 44, 2, 60}
```

*(gdb) step #--korak unutar fje findAndReturnMax, da smo koristili next citav poziv fje bi bio izvršen*

*findAndReturnMax (array1=0x76fff590, len=5, max=17) at max.c:20*

```
20          if(!array1 || (len <= 0)){
```

*(gdb) print array1[0]*

```
$3 = 17
```

*(gdb) p max*

```
$4 = 17
```

*(gdb) list*

```
15
16      int findAndReturnMax(int *array1, int len, int max){
17
```

```

18         int i;
19
20         if(!array1 || (len <= 0)){
21             return -1;
22         }
23
24         max = array1[0];

(gdb) list
25
26         for(i = 0; i < len; i++) {
27             if(max < array1[i]){
28                 max = array1[i];
29             }
30         }
31         return 0;
32     }

```

(gdb) break 26  
Breakpoint 2 at 0x40031c: file max.c, line 26.

(gdb) c  
Continuing.  
Breakpoint 2, findAndReturnMax (array1=0x76fff590, len=5, max=17) at max.c:26

```

26         for(i = 0; i < len; i++) {

```

(gdb) p i  
\$5 = 0  
(gdb) n  
27 if(max < array1[i]){

(gdb) display max **#-- display ispisuje varijablu svaki put kada se okine breakpoint**

```

1: max = 17
(gdb) display array1[i]
2: array1[i] = 17
(gdb) break 27
Breakpoint 3 at 0x40033c: file max.c, line 27.
(gdb) c
Continuing.

```

Breakpoint 3, findAndReturnMax (array1=0x76fff590, len=5, max=21) at max.c:27

```

27             if(max < array1[i]){
2: array1[i] = 44
1: max = 21

```

(gdb) c  
Continuing.

Breakpoint 3, findAndReturnMax (array1=0x76fff590, len=5, max=44) at max.c:27  
27                   if(max < array1[i]){  
2: array1[i] = 2  
1: max = 44

(gdb) c  
Continuing.

Breakpoint 3, findAndReturnMax (array1=0x76fff590, len=5, max=44) at max.c:27  
27                   if(max < array1[i]){  
2: array1[i] = 60  
1: max = 44

(gdb) n  
28                   max = array1[i];  
2: array1[i] = 60  
1: max = 44

(gdb) n  
30                   }  
2: array1[i] = 60  
1: max = 60 **#-- varijabla max = 60**

(gdb) where **#-- pokazuje stack frameove**

#0 findAndReturnMax (array1=0x76fff590, len=5, max=60) at max.c:30  
#1 0x0040027c in main (argc=1, argv=0x76fff6d4) at max.c:11

(gdb) frame 1 **#-- pomjeramo se u main context**

#1 0x0040027c in main (argc=1, argv=0x76fff6d4) at max.c:11  
11                   if( findAndReturnMax(arr, 5, max) == 0 ) {

(gdb) p max **#-- vidimo da je u mainu vrijednost varijable max = 17**  
\$6 = 17

(gdb) c  
Continuing.  
MAX: 17 **#-- main ispisuje vrijednost varijable max**

**#-- Ovo izgleda kao bug. findAndReturnMax nalazi najveću vrijednost ali je ne vraća u main funkciju. Da bismo popravili ovo trebamo proslijediti varijablu max po referenci ili vratiti vrijednost max varijable.**

## **Primjer 2:**

```
int main (int argc, char *argv[]) {  
  
    int a = 5;  
    int b = 76;  
    int c = a + b;  
}
```

Kompajliramo program, pokrecemo qemu i gdb:

```
gcc -target mips32r2el-linux -O0 -g saberi.c -o saberi
```

```
qemu-mipsel -g 1237 saberi &
```

```
gdb-multiarch saberi
```

*Reading symbols from saberi...done.*

*(gdb) target remt*

*Undefined target command: "remt". Try "help target".*

*(gdb) target remote:1237*

*Remote debugging using :1237*

*0x004001a0 in \_start ()*

*(gdb) break main*

*Breakpoint 1 at 0x400228: file saberi.c, line 2.*

*(gdb) c*

*Continuing.*

*Breakpoint 1, main (argc=1, argv=0x76fff6c4) at saberi.c:2*

*2 int a = 5;*

*(gdb) si #-- izvršava trenutnu instrukciju na koju pokazuje programski brojč*

*0x0040022c 2 int a = 5;*

*(gdb) x/i \$pc #-- prikazujemo jednu memorijsku jedinicu na kojoj se trenutno nalazi programski brojč*

*=> 0x40022c <main+24>: sw a0,8(s8)*

*(gdb) x/4i \$pc #-- prikazujemo 4 memorijske jedinice u odnosu na programski brojč*

*=> 0x40022c <main+24>: sw a0,8(s8)*

*0x400230 <main+28>: li a0,76*

*0x400234 <main+32>: sw a0,4(s8)*

*0x400238 <main+36>: li v0,0*

(gdb) info registers #-- *informacije o registrima*

```
zero  at  v0  v1  a0  a1  a2  a3
R0    00000000 00000000 00000000 00000000 00000005 76fff6c4 76fff6cc 00000000
      t0  t1  t2  t3  t4  t5  t6  t7
R8    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
      s0  s1  s2  s3  s4  s5  s6  s7
R16   00400214 76fff6c4 00000001 76fff6cc 00401000 00401000 00000000 00000000
      t8  t9  k0  k1  gp  sp  s8  ra
R24   00000000 00400214 00000000 00000000 00000000 76fff590 76fff590 004004a0
      sr  lo  hi  bad cause pc
      20000010 00000000 00000000 00000000 00000000 0040022c
      fsr  fir
      00000000 00739300
```

(gdb) si

3 int b = 76;

(gdb) p a

\$1 = 5

(gdb) p b

\$2 = 0

(gdb) p/x a #-- *ispis varijable a u heksadecimalnom obliku*

\$3 = 0x5

(gdb) si

0x00400234 3 int b = 76;

(gdb) si

0x00400238 3 int b = 76;

(gdb) si

4 int c = a + b;

(gdb) x/8i 0x00400234 #-- *prikazujemo 8 memorijskih jedinica u odnosu na memorijsku lokaciju 0x00400234*

```
0x400234 <main+32>: sw a0,4(s8)
0x400238 <main+36>: li v0,0
=> 0x40023c <main+40>: lw a0,8(s8)
0x400240 <main+44>: addiu a0,a0,76
0x400244 <main+48>: sw a0,0(s8)
0x400248 <main+52>: move sp,s8
0x40024c <main+56>: lw s8,20(sp)
0x400250 <main+60>: addiu sp,sp,24
```

(gdb) info registers

|     | zero     | at       | v0       | v1       | a0       | a1       | a2       | a3                |
|-----|----------|----------|----------|----------|----------|----------|----------|-------------------|
| R0  | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 0000004c | 76fff6c4 | 76fff6cc 00000000 |
|     | t0       | t1       | t2       | t3       | t4       | t5       | t6       | t7                |
| R8  | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000          |
|     | s0       | s1       | s2       | s3       | s4       | s5       | s6       | s7                |
| R16 | 00400214 | 76fff6c4 | 00000001 | 76fff6cc | 00401000 | 00401000 | 00000000 | 00000000          |
|     | t8       | t9       | k0       | k1       | gp       | sp       | s8       | ra                |
| R24 | 00000000 | 00400214 | 00000000 | 00000000 | 00000000 | 76fff590 | 76fff590 | 004004a0          |
|     | sr       | lo       | hi       | bad      | cause    | pc       |          |                   |
|     | 20000010 | 00000000 | 00000000 | 00000000 | 00000000 | 0040023c |          |                   |
|     | fsr      | fir      |          |          |          |          |          |                   |
|     | 00000000 | 00739300 |          |          |          |          |          |                   |

(gdb) x/4i \$pc

```
=> 0x400244 <main+48>:  sw    a0,0(s8)
0x400248 <main+52>:  move   sp,s8
0x40024c <main+56>:  lw     s8,20(sp)
0x400250 <main+60>:  addiu  sp,sp,24
```

(gdb) p c

\$8 = 0

(gdb) si

5 }

(gdb) p c

\$9 = 81

(gdb)

\$10 = 81

(gdb) c

Continuing.

[Inferior 1 (Remote target) exited normally]

(gdb) q

[1]+ Done qemu-mipsel -g 1237 saberi



