Local or global

main.c

```c
#include <stdio.h>

int x=3;

int main(){
    int x=5;
    printf("x = %d\n", x);
}
```

生成汇编代码

```
gcc -S main.c
```

```
    .file   "main.c"
    .text
    .globl  x
    .data
    .align 4
x:
    .long   3
    .def    __main; .scl    2;  .type   32; .endef
    .section .rdata,"dr"
.LC0:
    .ascii "x = %d\12\0"
    .text
    .globl  main
    .def    main;   .scl    2;  .type   32; .endef
    .seh_proc   main
main:
    pushq   %rbp
    .seh_pushreg    %rbp
    movq    %rsp, %rbp
    .seh_setframe   %rbp, 0
    subq    $48, %rsp
    .seh_stackalloc 48
    .seh_endprologue
```

```
    call    __main
    movl    $5, -4(%rbp)
    movl    -4(%rbp), %eax
    movl    %eax, %edx
    leaq    .LC0(%rip), %rcx
    call    printf
    movl    $0, %eax
    addq    $48, %rsp
    popq    %rbp
    ret
    .seh_endproc
    .ident  "GCC: (GNU) 10.2.0"
    .def    printf; .scl    2;  .type   32; .endef
```

汇编, 链接, 运行

```
gcc -c main.s
gcc main.o
.\a.exe
```

得到输出

```
x = 5
```

在汇编代码中可以看到, 局部变量`x`被放到了栈上

```
movl    $5, -4(%rbp)
```

而全局变量`x`被放到了rdata段

```
x:
    .long   3
    .def    __main; .scl    2;  .type   32; .endef
    .section .rdata,"dr"
```

输出的关键部分在第26行到第29行

```
movl    -4(%rbp), %eax
movl    %eax, %edx
leaq    .LC0(%rip), %rcx
call    printf
```

这里通过`rbp`取了栈上的局部变量(`x`), 作为`printf()`的参数, 要想让程序输出全局变量的`x`, 只要把这里改为取全局的`x`即可

```
movl    x(%rip), %eax
movl    %eax, %edx
leaq    .LC0(%rip), %rcx
call    printf
```

修改后的完整代码

```
    .file   "main.c"
    .text
    .globl  x
    .data
    .align 4
x:
    .long   3
    .def    __main; .scl    2;  .type   32; .endef
    .section .rdata,"dr"
.LC0:
    .ascii "x = %d\12\0"
    .text
    .globl  main
    .def    main;   .scl    2;  .type   32; .endef
    .seh_proc   main
main:
    pushq   %rbp
    .seh_pushreg    %rbp
    movq    %rsp, %rbp
    .seh_setframe   %rbp, 0
    subq    $48, %rsp
    .seh_stackalloc 48
    .seh_endprologue
    call    __main
    movl    $5, -4(%rbp)
    movl    x(%rip), %eax
    movl    %eax, %edx
    leaq    .LC0(%rip), %rcx
    call    printf
    movl    $0, %eax
    addq    $48, %rsp
    popq    %rbp
    ret
```

```
    .seh_endproc
    .ident  "GCC: (GNU) 10.2.0"
    .def    printf; .scl    2;  .type   32; .endef
```

重新汇编, 链接, 运行

```
gcc -c main.s
gcc main.o
.\a.exe
```

得到运行结果

```
x = 3
```