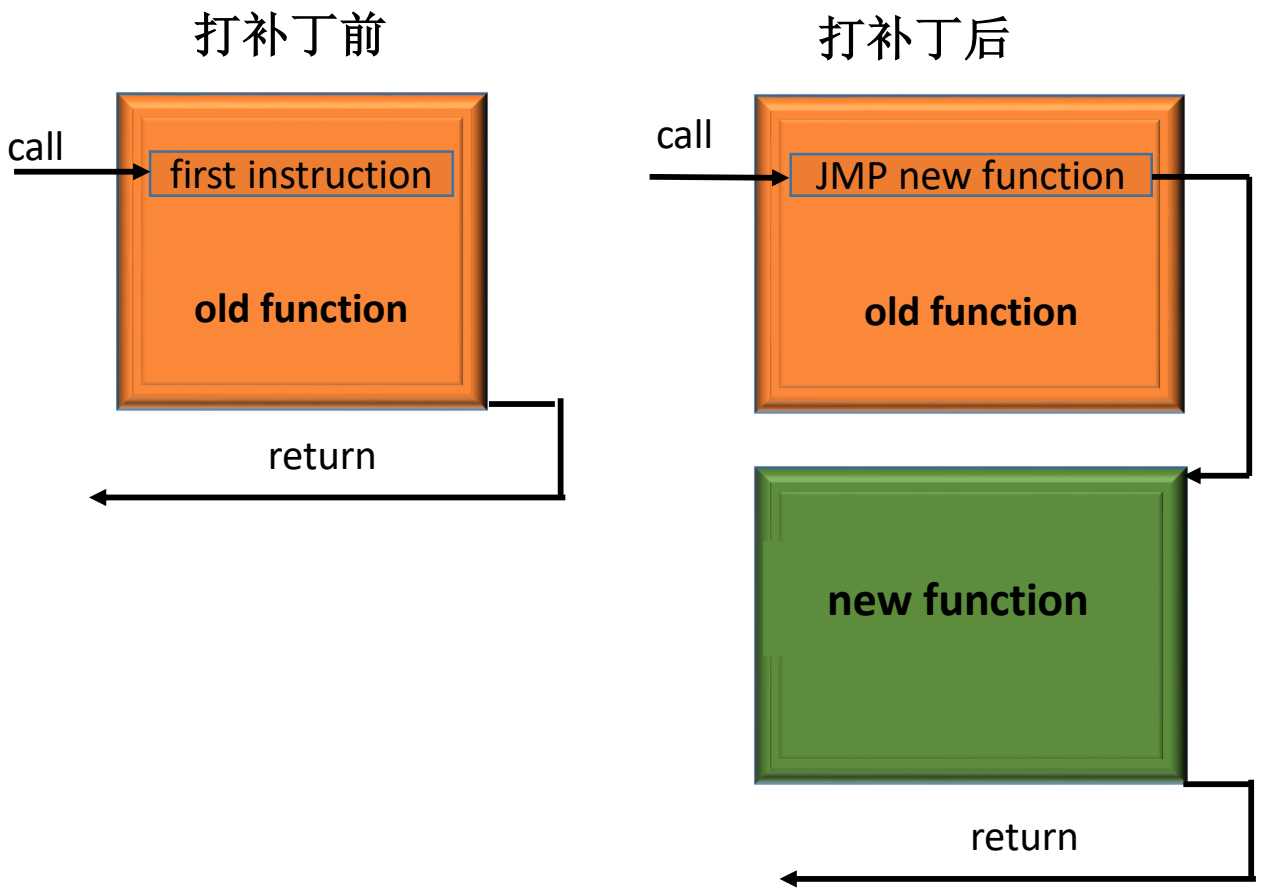
云环境动态修复技术实践

刘琦

基本原理



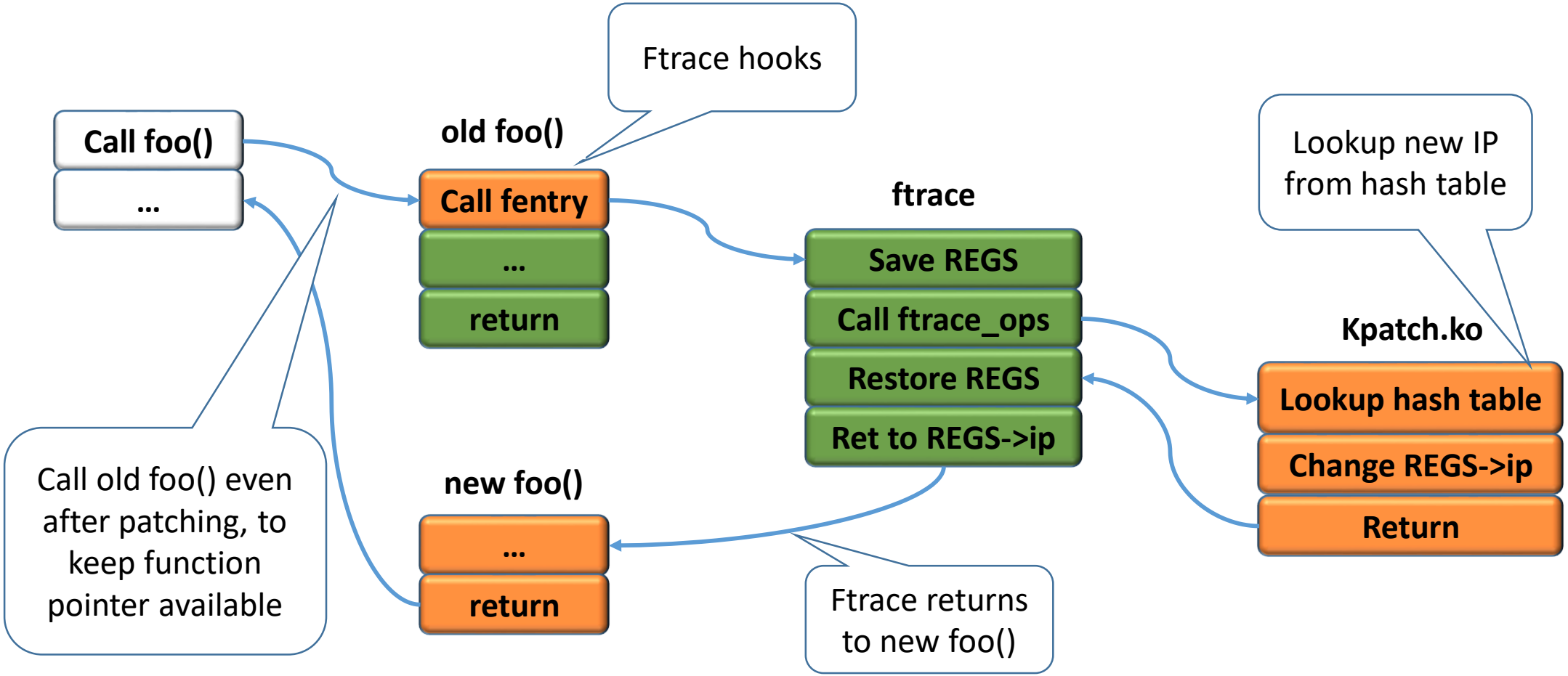
优点

- 支持C/汇编程序
- 对程序无特殊要求
- 上线成本低

缺点

- 不能100%覆盖全部文件

内核热补丁的原理


Ftrace hooks
Call foo()
...
old foo()
Call fentry
...
return
ftrace
Save REGS
Call ftrace_ops
Restore REGS
Ret to REGS->ip
Lookup new IP from hash table
Kpatch.ko
Lookup hash table
Change REGS->ip
Return
Call old foo() even after patching, to keep function pointer available
new foo()
...
return
Ftrace returns to new foo()

用户热补丁的挑战

- 无类似ftrace的机制。
- 会调用外部API，需解析PLT & GOT。
- 使用TLS数据，需对linker的优化结果进行逆向。
- 使用扩展指令集，需要更健壮的decoder。

编译时的要求(--ffunction-sections)

without --function-sections

```
0000000000000010 <main>:
  10:   55                      push   %rbp
  11:   48 89 e5                mov    %rsp,%rbp
  14:   48 83 ec 10             sub    $0x10,%rsp
  18:   89 7d fc                mov    %edi,-0x4(%rbp)
  1b:   48 89 75 f0             mov    %rsi,-0x10(%rbp)
  1f:   b8 00 00 00 00          mov    $0x0,%eax
  24:   e8 d7 ff ff ff          callq  0 <foo>
  29:   c9                      leaveq
  2a:   c3                      retq
```

立即数，相对跳转至foo

with --ffunction-sections

```
0000000000000000 <main>:
   0:   55                      push   %rbp
   1:   48 89 e5                mov    %rsp,%rbp
   4:   48 83 ec 10             sub    $0x10,%rsp
   8:   89 7d fc                mov    %edi,-0x4(%rbp)
   b:   48 89 75 f0             mov    %rsi,-0x10(%rbp)
   f:   b8 00 00 00 00          mov    $0x0,%eax
  14:   e8 00 00 00 00          callq  19 <main+0x19>
  19:   c9                      leaveq
  1a:   c3                      retq
```

重定位，指向foo()

```
Relocation section '.rela.text.main' at offset 0x6f8 contains 1 entries
  Offset          Info           Type           Sym. Value    Sym. Name
000000000015  000700000002 R_X86_64_PC32     0000000000000000 .text.foo
```

符号的唯一性

symtab

```
 41: 0000000000000000     0 FILE    LOCAL  DEFAULT  ABS foo.c
 42: 0000000000600884     4 OBJECT  LOCAL  DEFAULT   24 var
 43: 0000000000400474    16 FUNC    LOCAL  DEFAULT   13 foo
 44: 0000000000000000     0 FILE    LOCAL  DEFAULT  ABS main.c
 45: 0000000000600888     4 OBJECT  LOCAL  DEFAULT   24 var
 46: 0000000000400484    16 FUNC    LOCAL  DEFAULT   13 foo
```

scope foo.c

scope main.c

```
static int var  =   2;

static void foo()
{
    var =   0;
}
int main(int argc, char** argv)
{
    foo();
}
"main.c" 38 lines --73%--
```

```
static int var  =   2;

static void foo()
{
    var =   2;
}


"foo.c" 36 lines --86%--
```

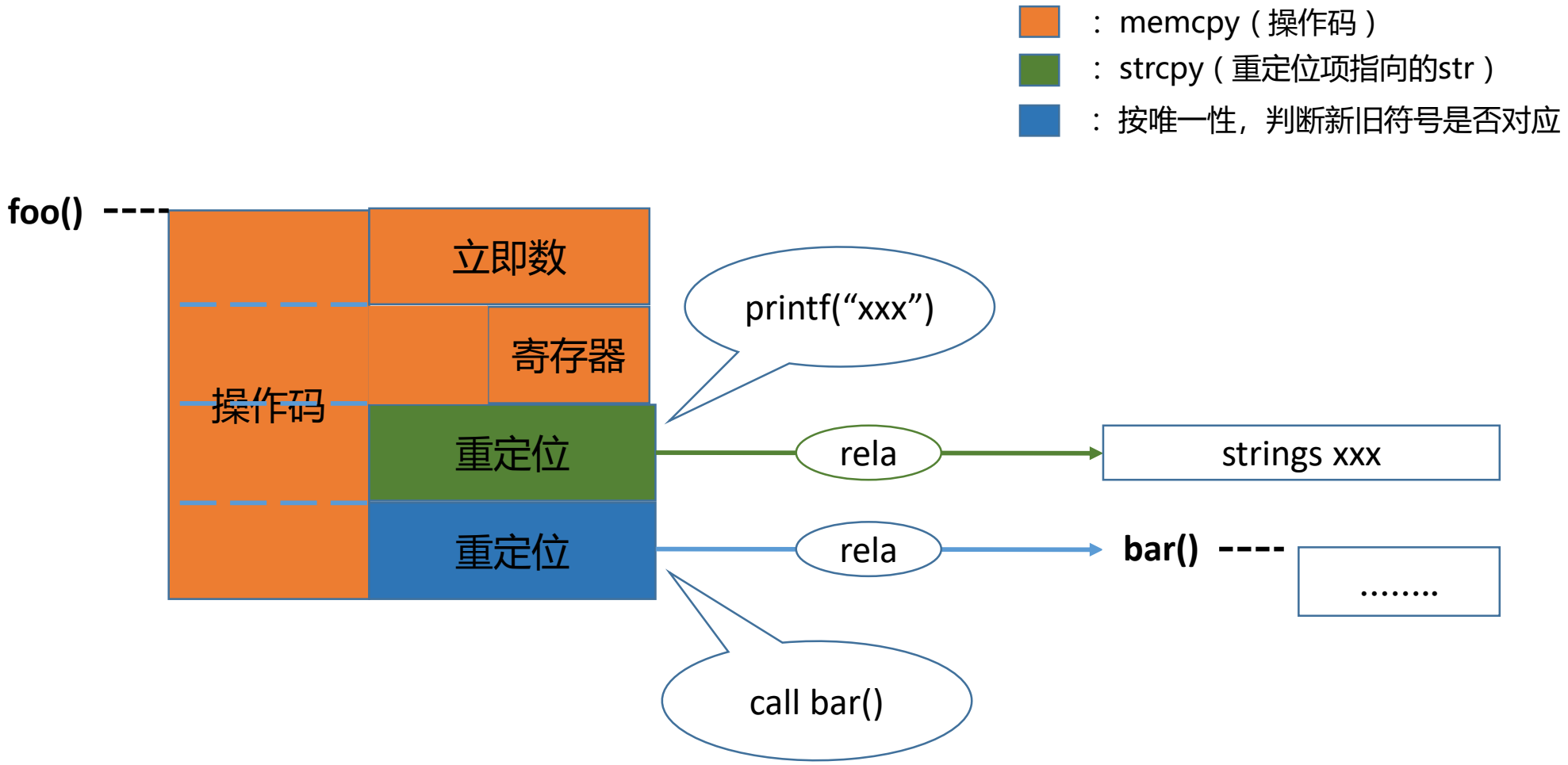
文件的唯一标识

- FILE名称
- FILE中变量和函数名称的集合

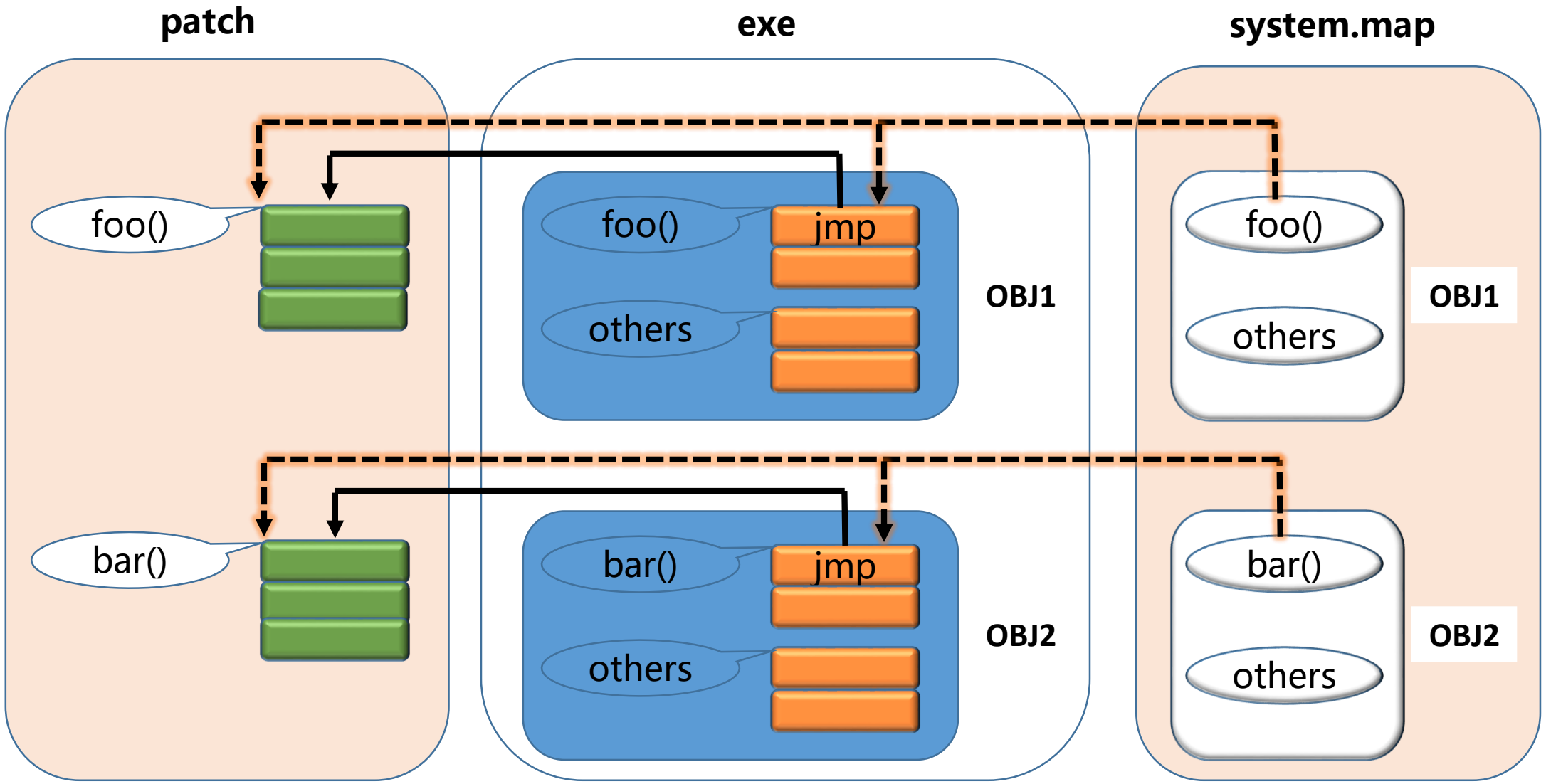
函数的唯一标识

- 函数名称

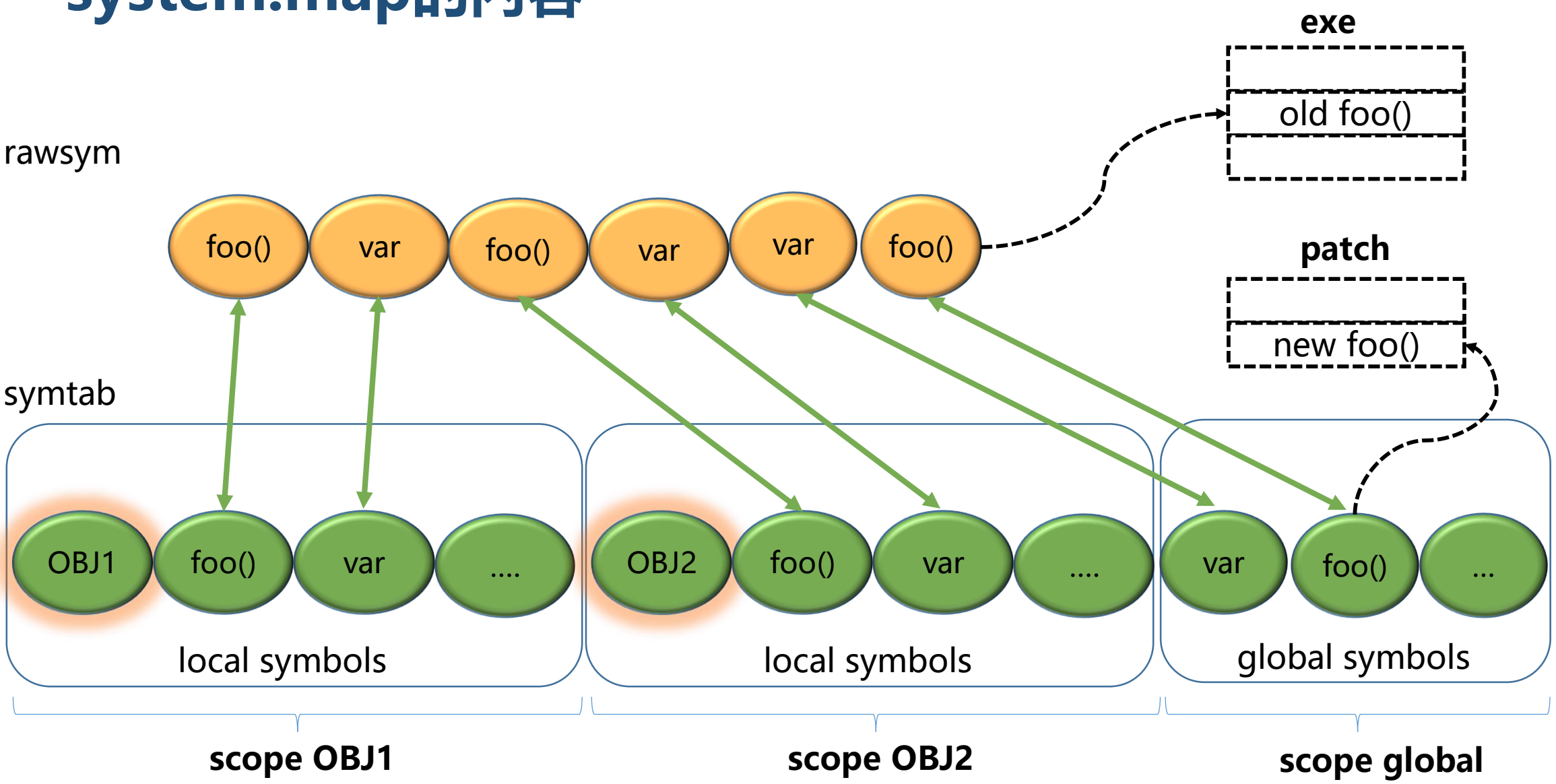
生成补丁的原理


：memcpy（操作码）
：strcpy（重定位项指向的str）
：按唯一性，判断新旧符号是否对应
foo() ----
立即数
寄存器
操作码
重定位
重定位
printf(“xxx”)
rela
strings xxx
rela
bar() ----
……..
call bar()

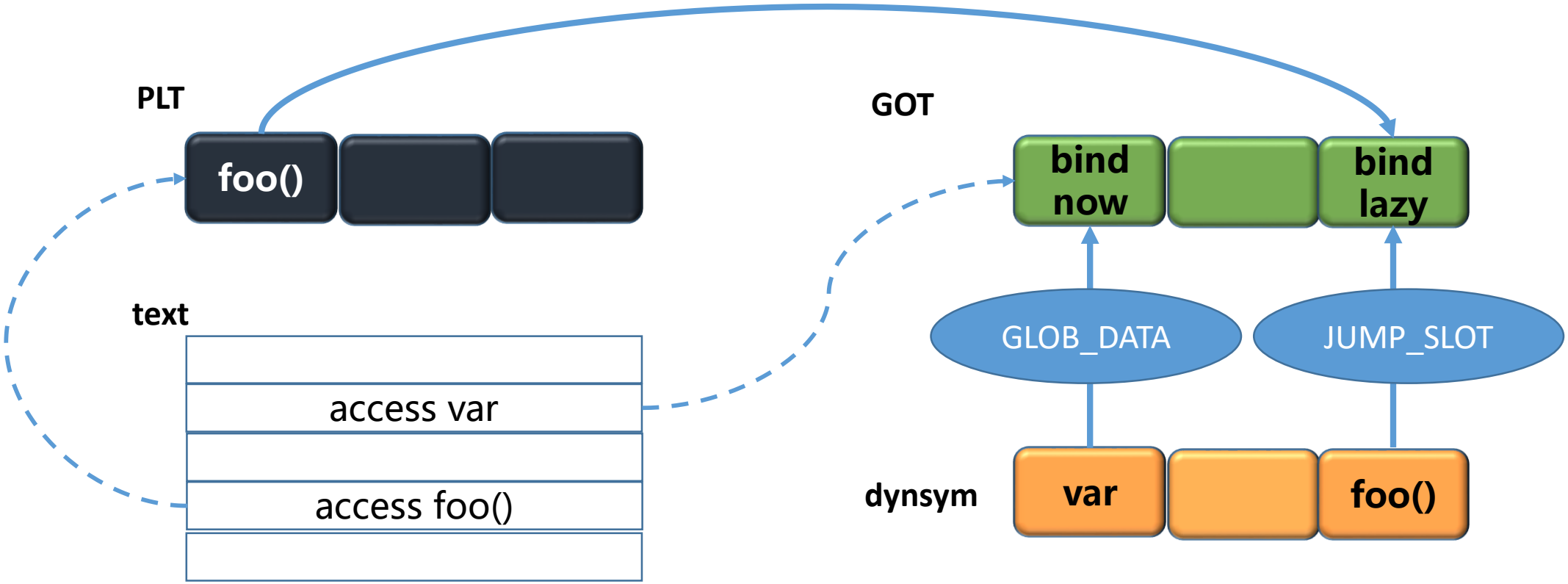
打入补丁的原理


patch
exe
system.map
foo()
jmp
foo()
others
OBJ1
foo()
others
OBJ1
bar()
jmp
bar()
others
OBJ2
bar()
others
OBJ2

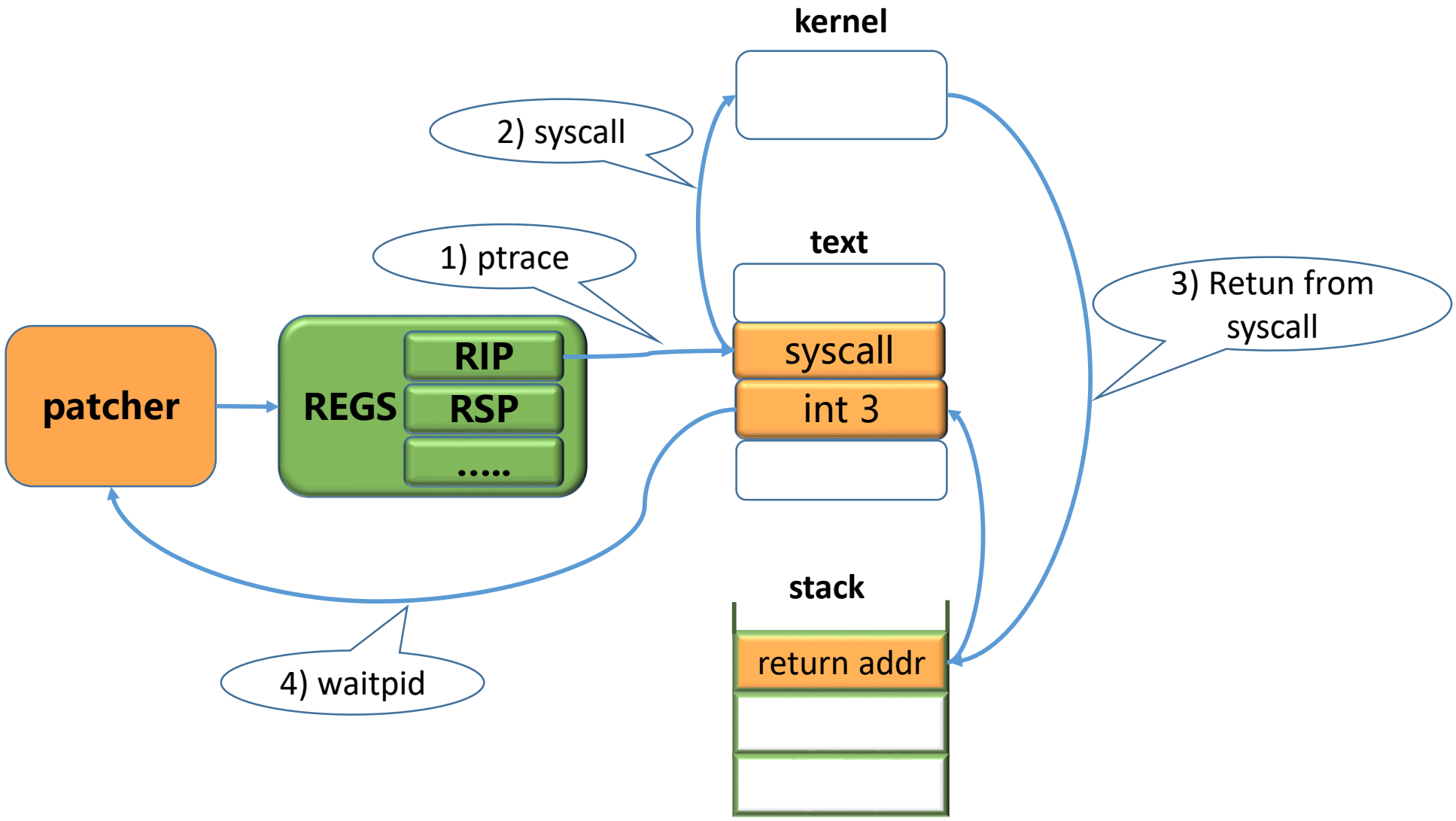
system.map的内容



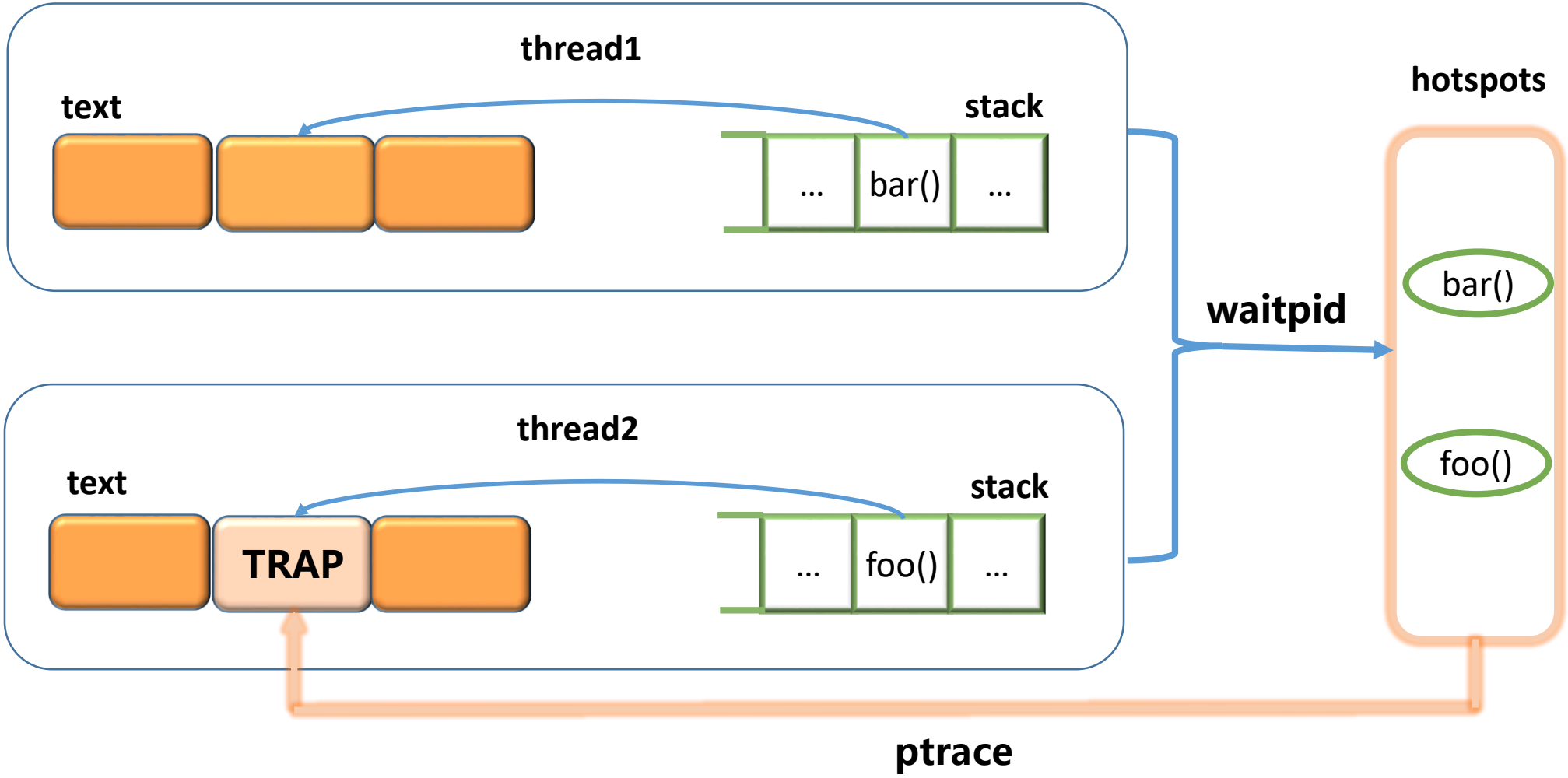
GOT & PLT的解析


PLT
foo()
text
access var
access foo()
GOT
bind now
bind lazy
GLOB_DATA
JUMP_SLOT
dynsym
var
foo()

syscall(mmap, open...)



热点函数


thread1
text
stack
...
bar()
...
thread2
text
stack
TRAP
...
foo()
...
hotspots
bar()
foo()
waitpid
ptrace

问题一: 非定长指令

解析相对跳转指令的目标地址(RIP = next instruction)，需x86的decoder。

Porting from:

- kpatch

  为解析相对跳转，kpatch实现了x86标准指令集的decoder。

- kvm

  为解析MMIO指令，kvm实现了更多指令集的decoder。

问题二: 语法块

```c
void foo()
{
    static int aaa  =   1;

    aaa =   0;
}

int main(int argc, char** argv)
{
    static int aaa  =   1;

    aaa =   0;
}
```

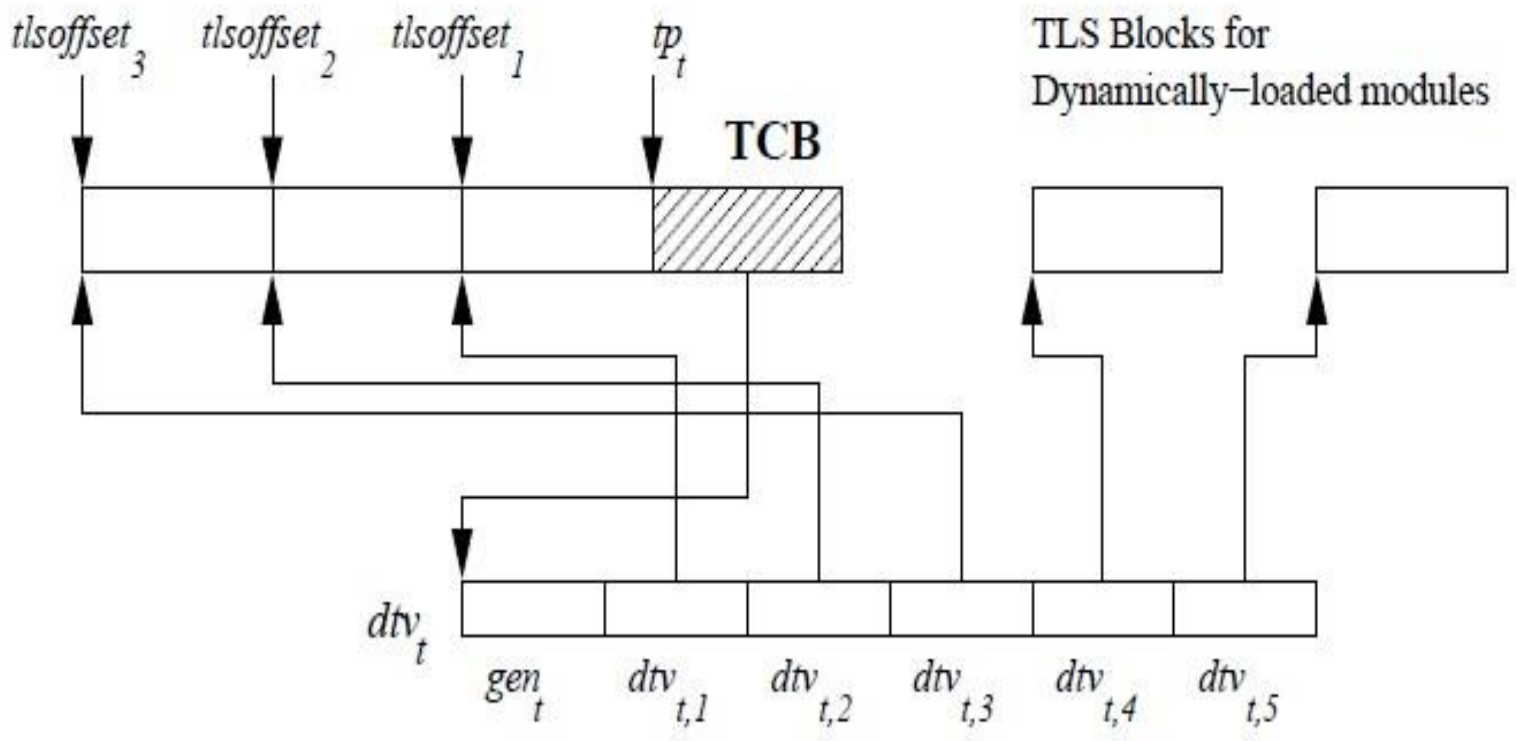
```
Symbol table '.symtab' contains 16 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000000000     0 FILE    LOCAL  DEFAULT  ABS test.c
     2: 0000000000000000     0 SECTION LOCAL  DEFAULT    1
     3: 0000000000000000     0 SECTION LOCAL  DEFAULT    2
     4: 0000000000000000     0 SECTION LOCAL  DEFAULT    3
     5: 0000000000000000     0 SECTION LOCAL  DEFAULT    4
     6: 0000000000000000     4 OBJECT  LOCAL  DEFAULT    9 aaa.1593
     7: 0000000000000000     0 SECTION LOCAL  DEFAULT    6
     8: 0000000000000000     4 OBJECT  LOCAL  DEFAULT    8 aaa.1598
     9: 0000000000000000     0 SECTION LOCAL  DEFAULT    8
    10: 0000000000000000     0 SECTION LOCAL  DEFAULT    9
    11: 0000000000000000     0 SECTION LOCAL  DEFAULT   11
    12: 0000000000000000     0 SECTION LOCAL  DEFAULT   12
    13: 0000000000000000     0 SECTION LOCAL  DEFAULT   10
    14: 0000000000000000    16 FUNC    GLOBAL DEFAULT    4 foo
    15: 0000000000000000    23 FUNC    GLOBAL DEFAULT    6 main
```

```
Relocation section '.rela.text.foo' at offset 0x778 contains 1 entries:
  Offset          Info           Type           Sym. Value    Sym. Name + Addend
000000000006  000a00000002 R_X86_64_PC32     0000000000000000 .data.aaa.1593 - 8

Relocation section '.rela.text.main' at offset 0x790 contains 1 entries:
  Offset          Info           Type           Sym. Value    Sym. Name + Addend
00000000000d  000900000002 R_X86_64_PC32     0000000000000000 .data.aaa.1598 - 8
```

- 以重定位项，来区分同文件中，同名的静态变量。

问题三: 修饰符static __thread

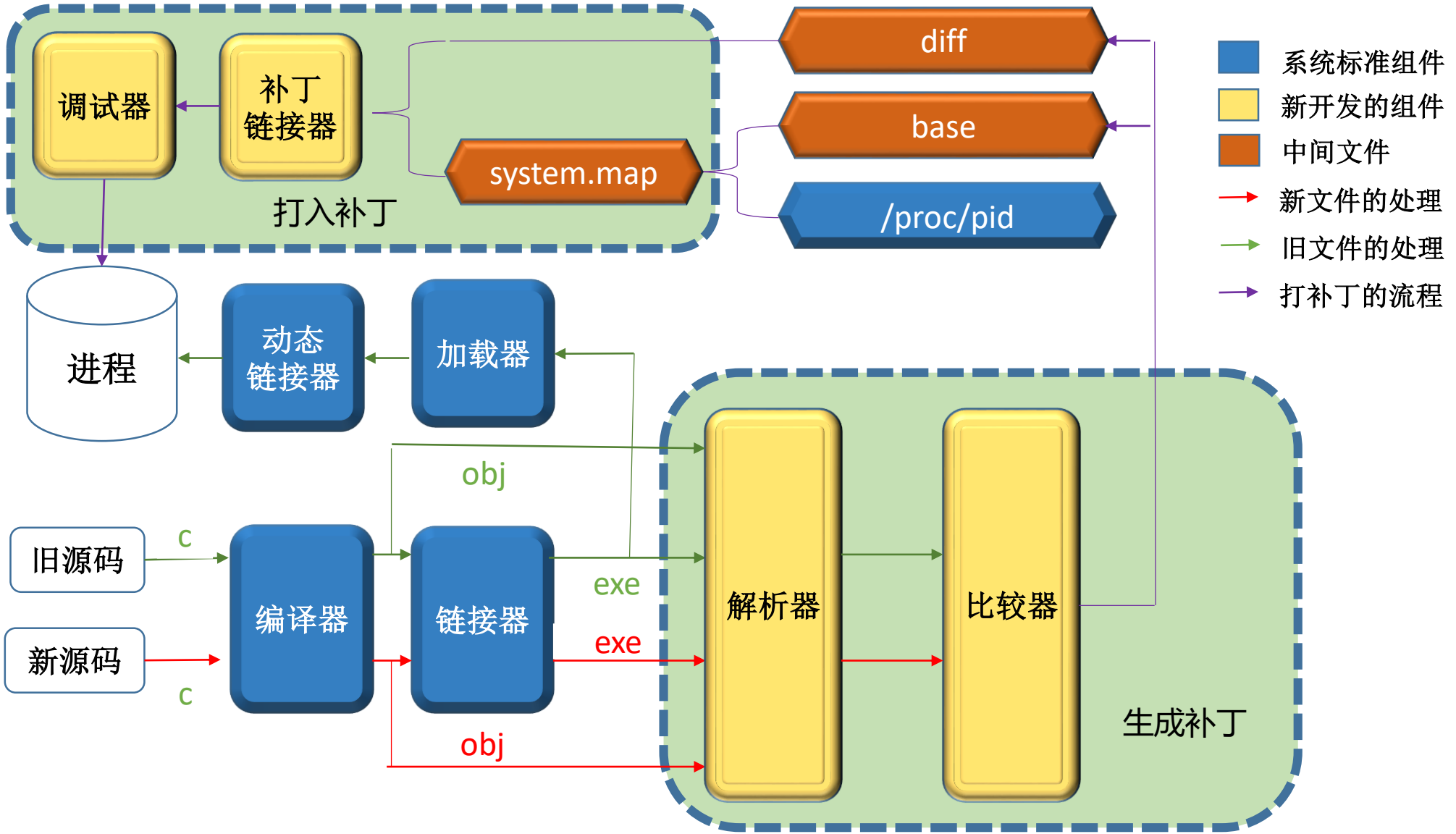


链接器生成操作码

- GD：针对dlopen的库
- LD ：针对dlopen的库
- LE ：针对依赖的库
- IE ：针对EXE本身

整体方案

可运维性


调试器
补丁
链接器
system.map
打入补丁
diff
base
/proc/pid
系统标准组件
新开发的组件
中间文件
新文件的处理
旧文件的处理
打补丁的流程
进程
动态
链接器
加载器
obj
旧源码
c
新源码
c
编译器
链接器
exe
exe
obj
解析器
比较器
生成补丁

示例

补丁内容

```
Section Headers:
  [Nr] Name              Type            Address          Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            0000000000000000 000000 000000 00   0   0   0
  [ 1] .shstrtab         STRTAB          0000000000000000 000270 000181 00      0   0   1
  [ 2] .text             PROGBITS        0000000000000000 001000 0000d0 00 AX   0   0  16
  [ 3] .qpatch.symtab.dst SYMTAB         0000000000000000 001100 0005d8 18      4   0   8
  [ 4] .qpatch.strtab.dst STRTAB         0000000000000000 001700 0a1c41 00      0   0   1
  [ 5] .qpatch.symtab.src SYMTAB         0000000000000000 0a3350 0005d8 18      6   0   8
  [ 6] .qpatch.strtab.src STRTAB         0000000000000000 0a3930 0a1c41 00      0   0   1
  [ 7] .qpatch.diffdata  RELA            0000000000000000 145800 0005d8 18      6   3   8
```

打入结果

```
7f667a9ae000-7f667a9af000 rw-p 00000000 00:00 0
7f667a9af000-7f667a9b0000 r-xp 00000000 00:00 0
7f667a9b7000-7f667ac90000 r-xp 00000000 08:02 2071328      /usr/libexec/qemu-kvm (deleted)
7f667ae8f000-7f667af55000 r--p 002d8000 08:02 2071328      /usr/libexec/qemu-kvm (deleted)
7f667af55000-7f667af81000 rw-p 0039e000 08:02 2071328      /usr/libexec/qemu-kvm (deleted)
7f901ca3a000-7f901ca3c000 r--p 00000000 08:02 2263406      /home/liuqi16/tmp/qemu-kvm.metadata
```

Bai du 百度

谢谢！

Thank you