openEuler内核热补丁介绍

目录

CONTENT

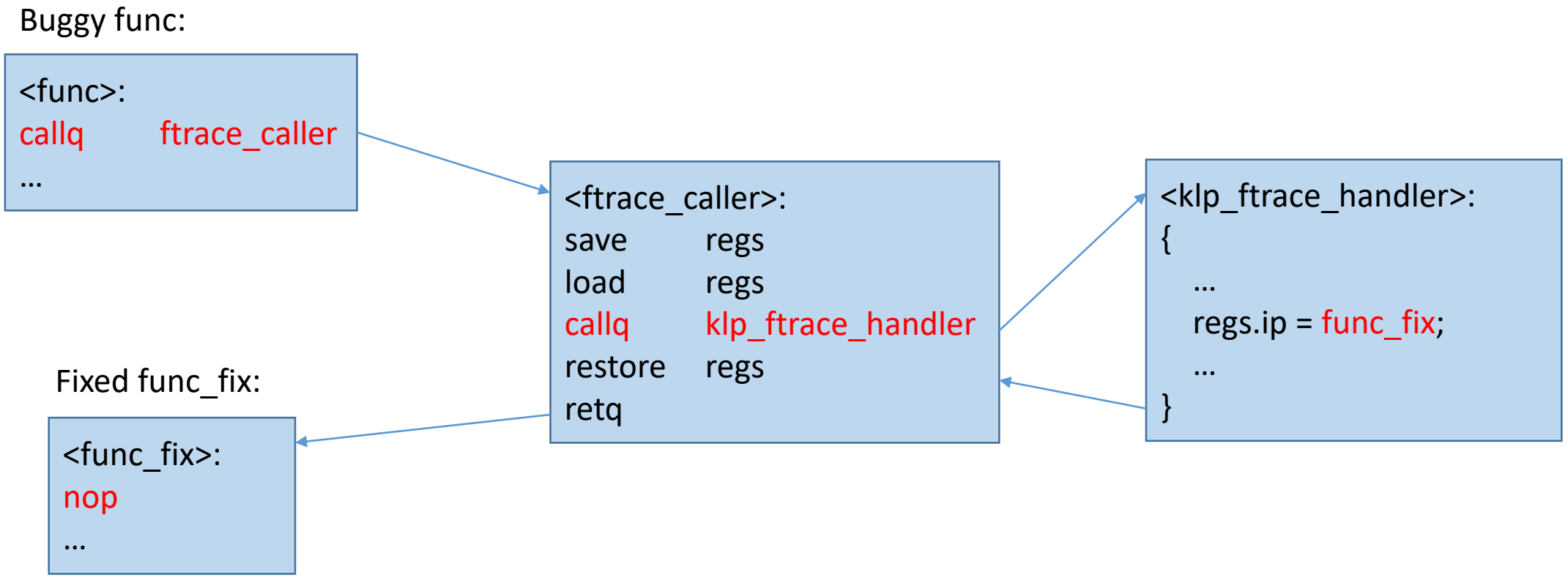
OpenEuler

Livepatch介绍

Livepatch即为内核热补丁，通常在系统不可重启的情况下，用于修复内核以及内核模块的函数bug

	Linux社区方案	openEuler方案
实现方法	基于ftrace跳转	直接修改指令跳转
支持的架构	X86、ppc64	X86、arm64、arm32、ppc64、ppc32
生效时间	延后生效	立即生效
是否暂停业务	否	是（stop_machine）
是否与ftrace兼容	是	否
是否支持修改notrace函数	否	是
运行性能	略差	优
是否支持重新激活	否	是
是否insmod时激活	是	否

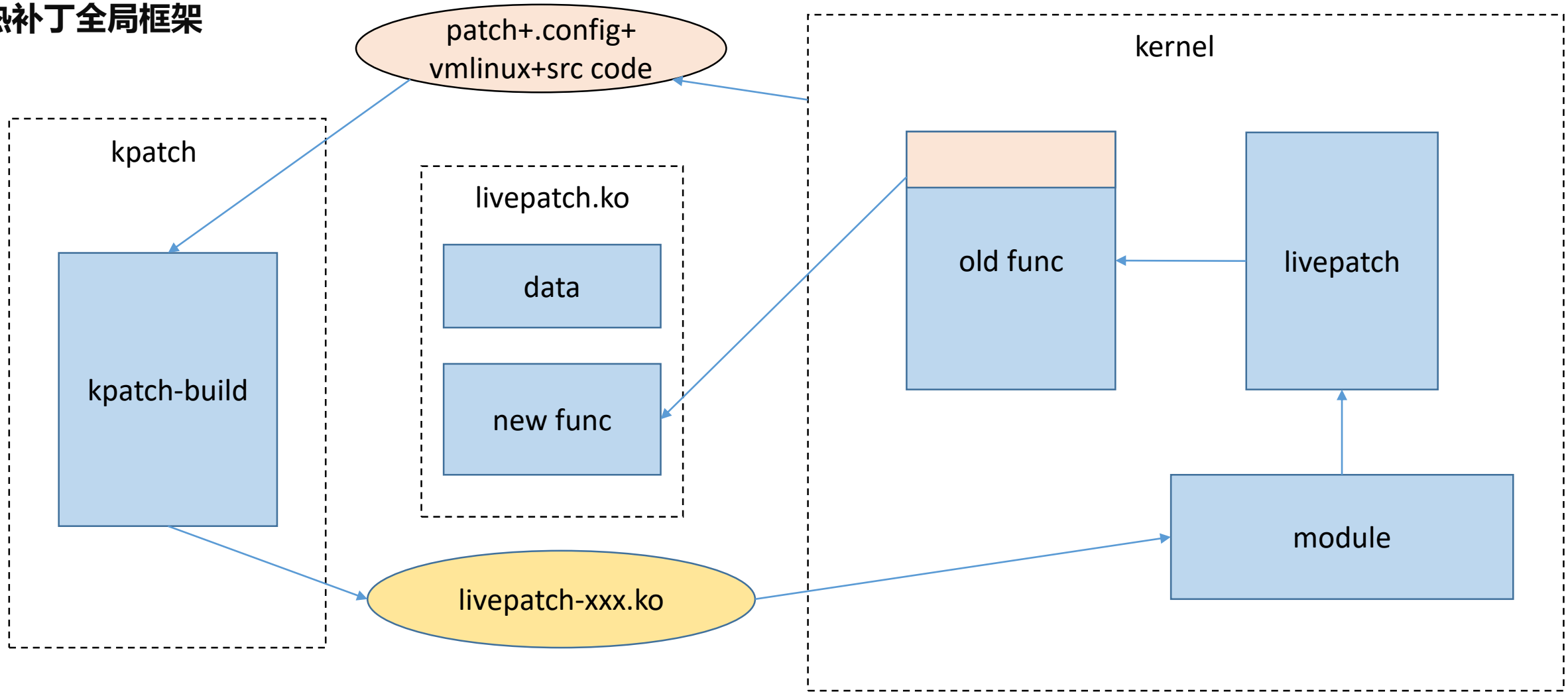
Livepatch介绍



OpenEuler

Livepatch框架

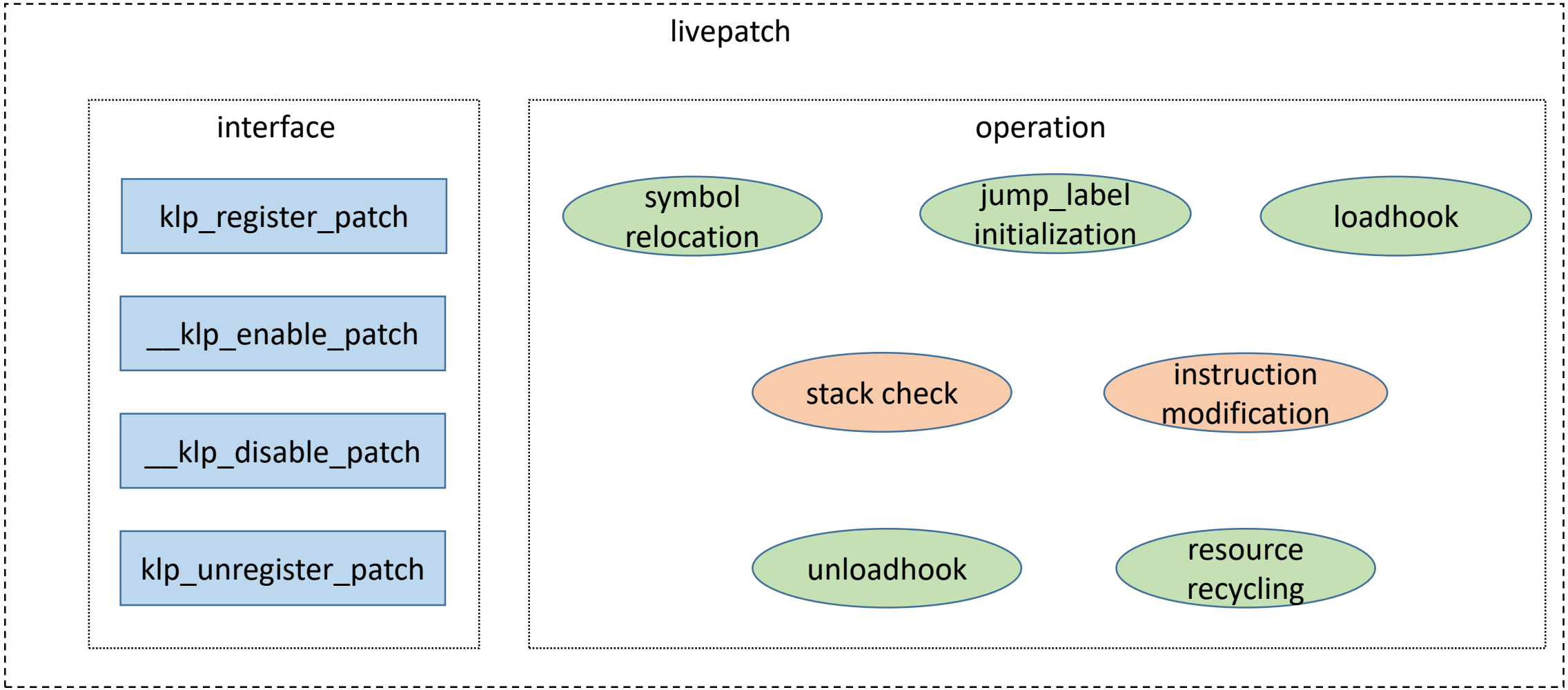
热补丁全局框架



OpenEuler

Livepatch框架

热补丁内部框架



OpenEuler

Livepatch框架

核心数据结构

klp_func

old_name 函数名称
new_func 热补丁函数地址
old_sympos 同名函数下标
old_func 原始函数地址
node 热补丁函数链表
stack_node 该函数热补丁链表
old_size 原始函数长度
new_size 热补丁函数长度
patched 生效与否
func_node 临时数据结构

klp_object

name 模块名
funcs klp_func数组首地址
hooks_load loadhook数组首地址
hooks_unload unloadhook数组首地址
patched 生效与否
mod 对应模块结构

Livepatch框架

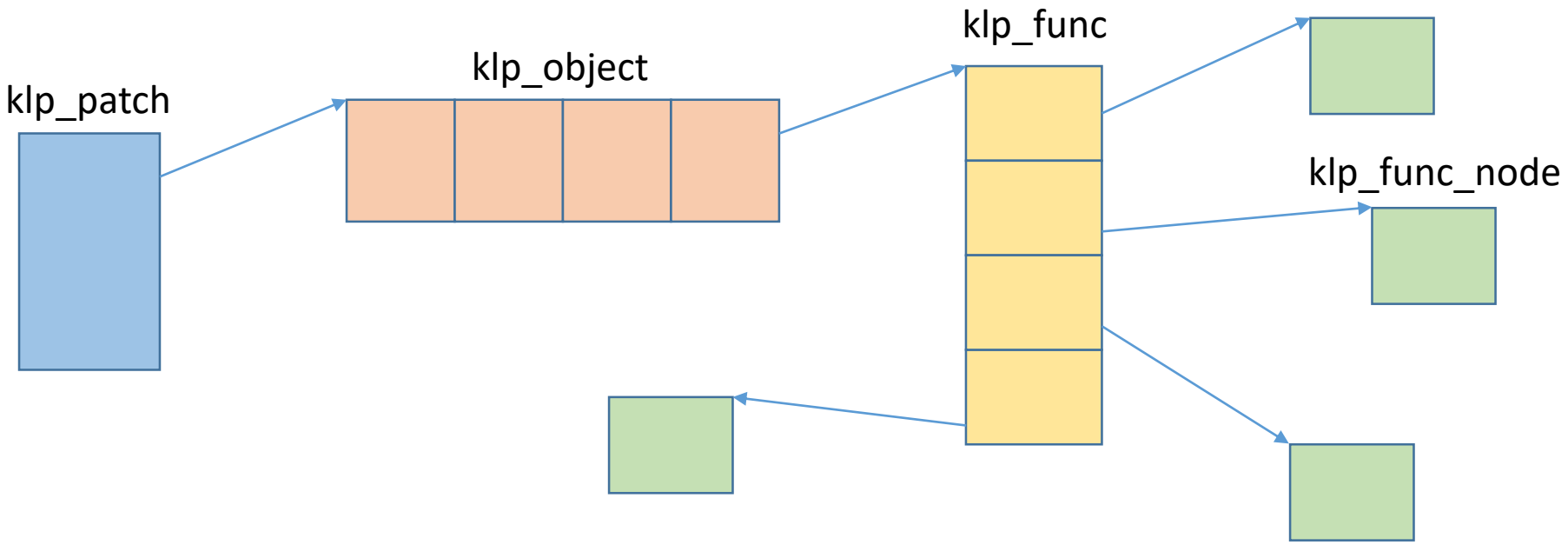
核心数据结构

klp_patch

mod 热补丁模块
objs klp_object数组首地址
states patch的状态
list 热补丁链表

klp_func_node(x86)

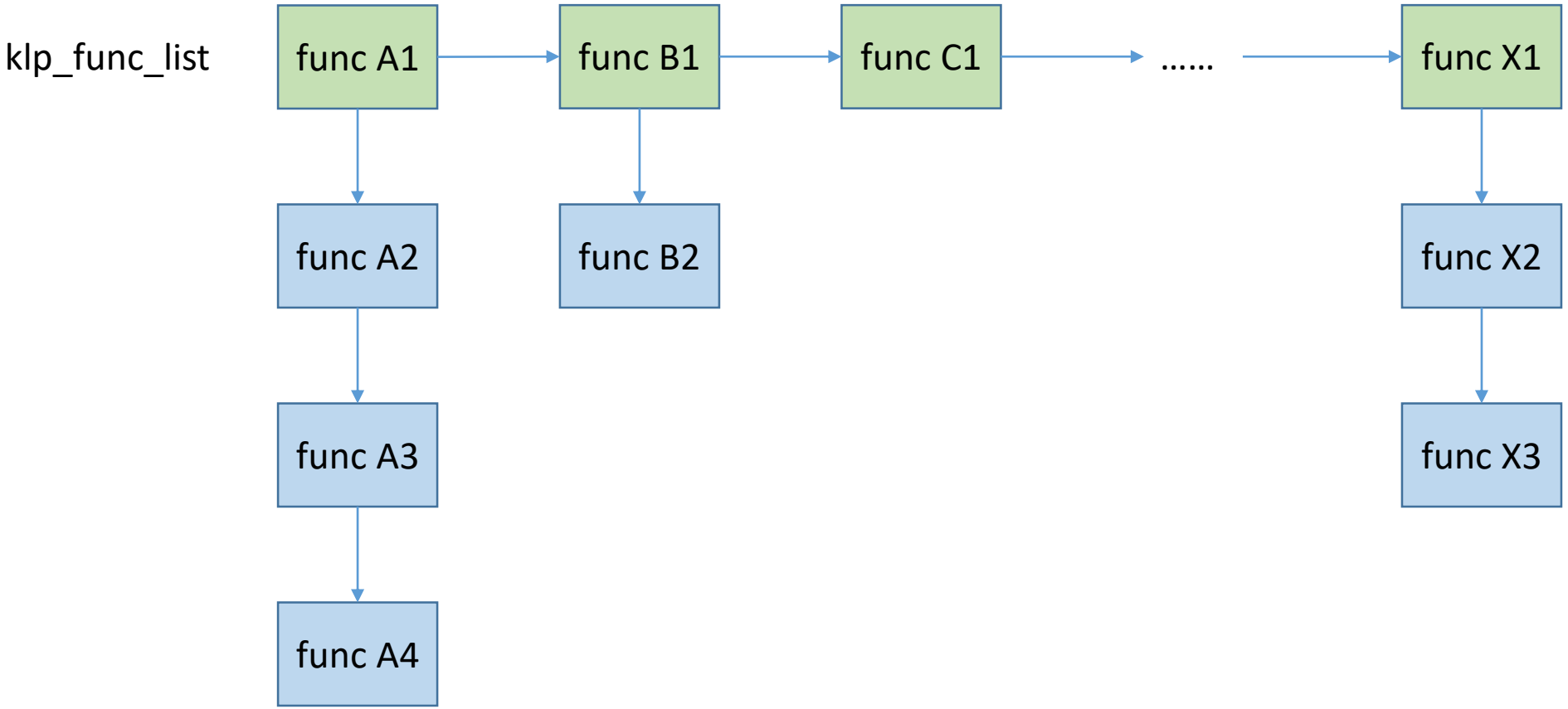
node klp_func_node链表结点
func_stack 同个函数的多个结点
old_func 原始函数地址
old_code 替换区的指令备份



OpenEuler

Livepatch框架

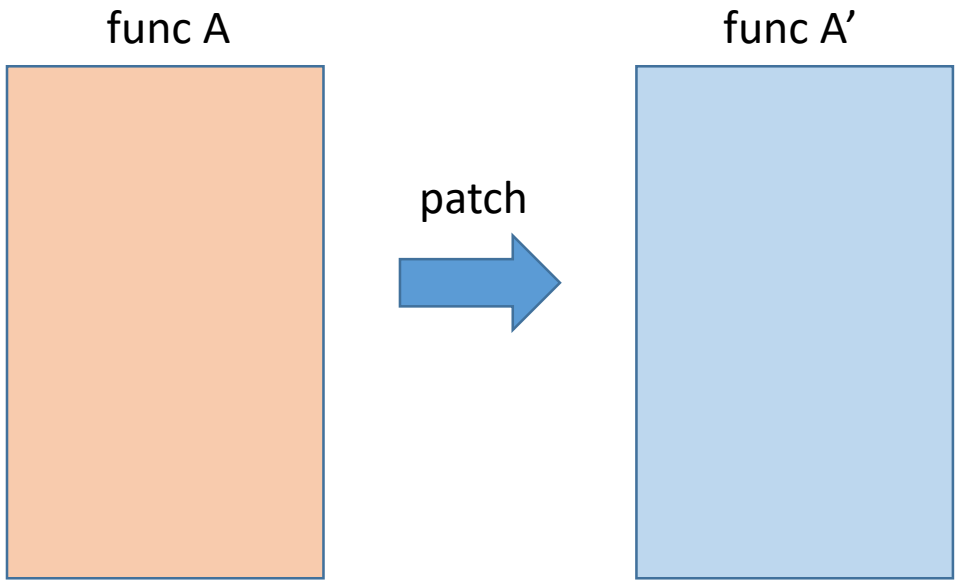
热补丁函数管理



OpenEuler

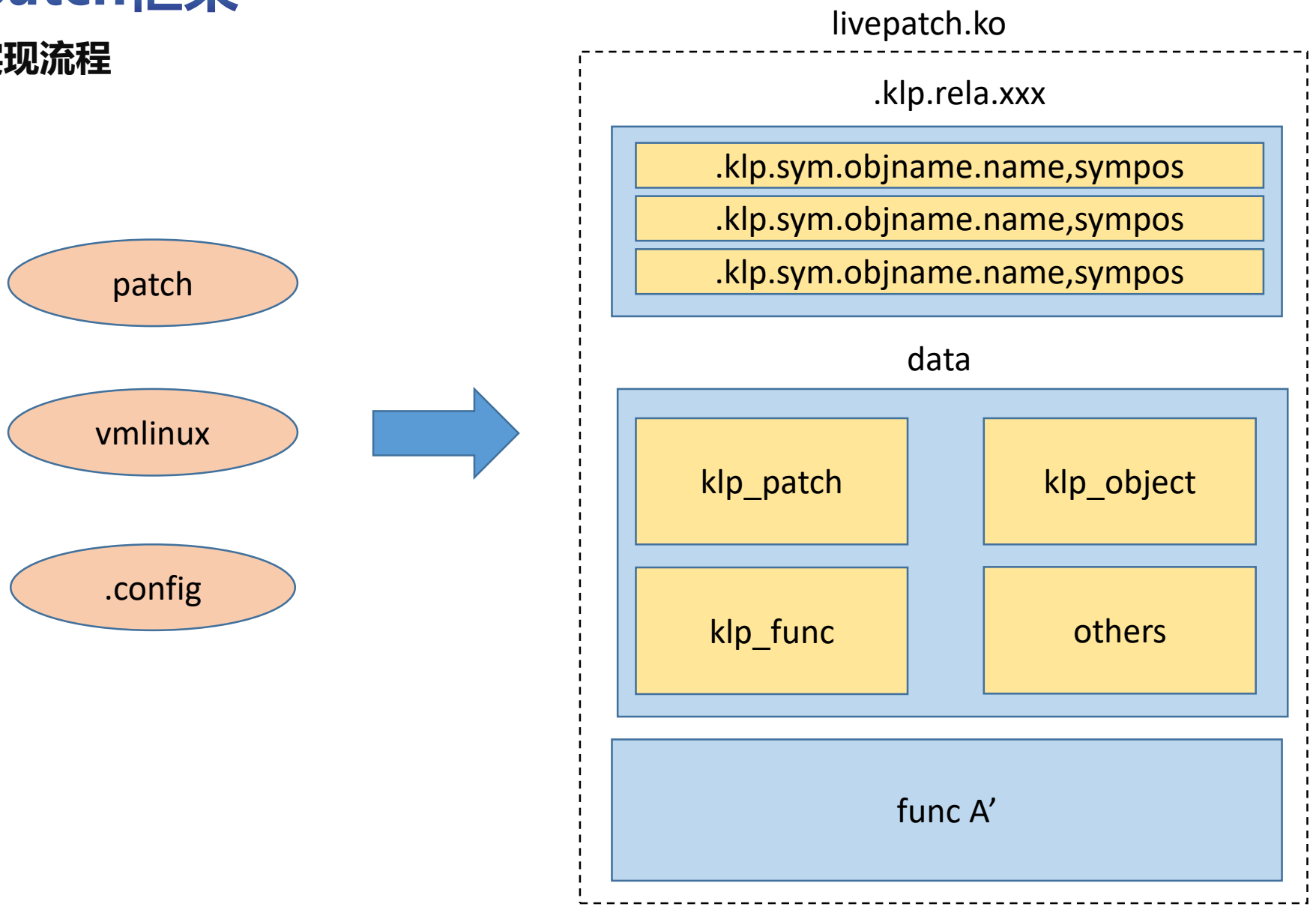
Livepatch框架

热补丁实现流程



Livepatch框架

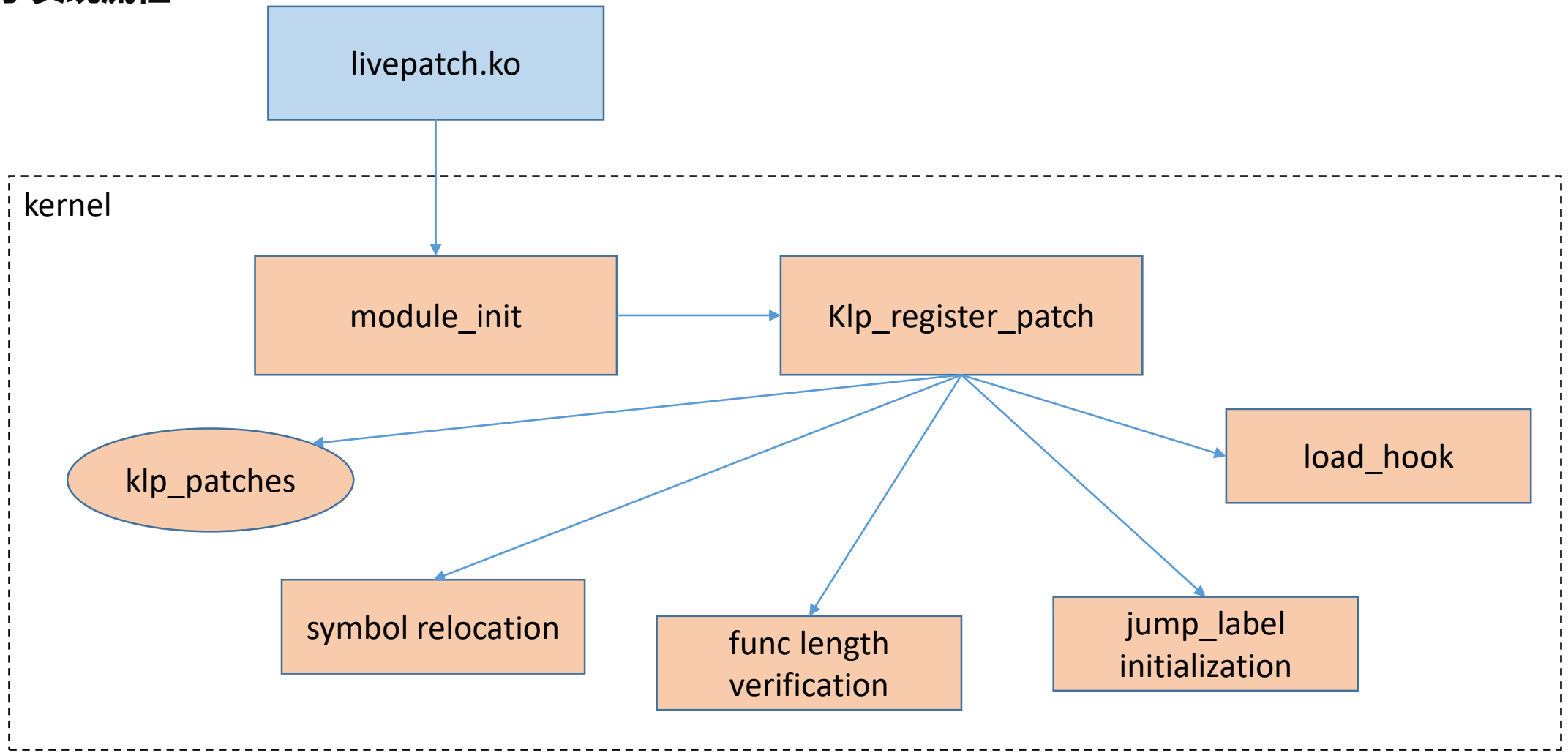
热补丁实现流程



OpenEuler

Livepatch框架

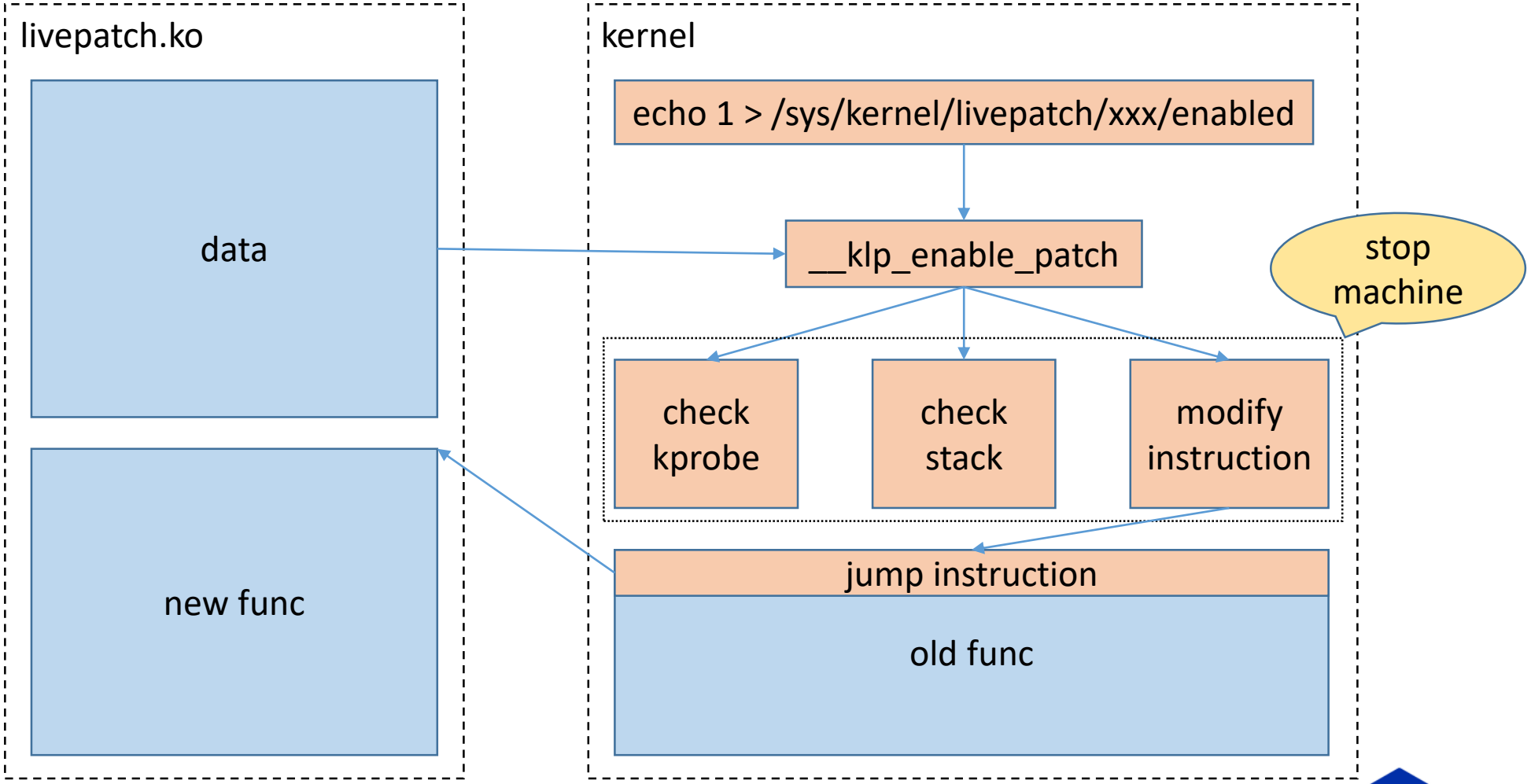
热补丁实现流程



OpenEuler

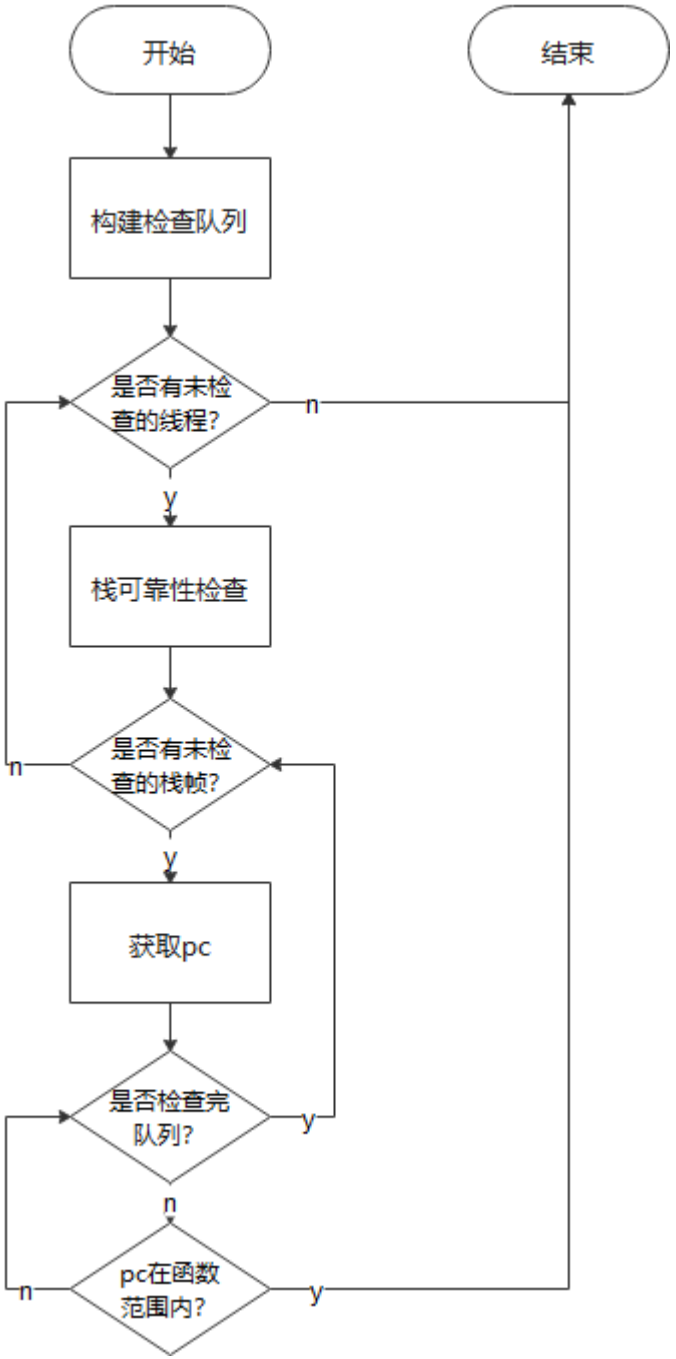
Livepatch框架

热补丁实现流程



OpenEuler

栈检查



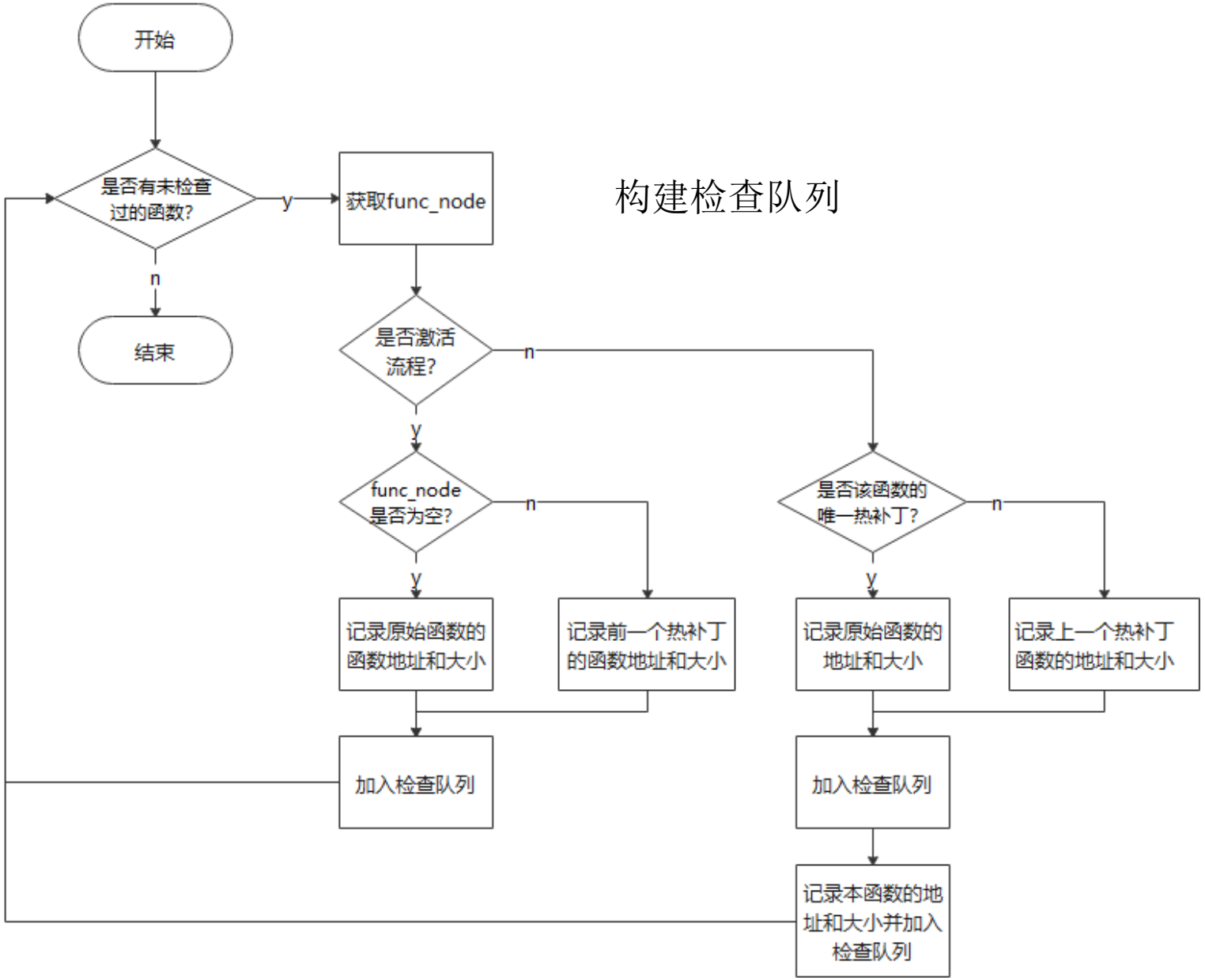
热补丁栈检查的目的是为了保证代码修改时，对应的函数（或指令）没有被运行。栈检查主要有两个流程会调用，热补丁激活前和去使能前，都是在stop_machine内执行的。

若没有执行栈检查，运行了部分修改指令，会跳转至未知区域，导致crash。

```
356 #ifdef CONFIG_ARCH_STACKWALK
357 >-------->-------ret = stack_trace_save_tsk_reliable(t, trace_entries, MAX_STACK_ENTRIES);
358 >-------->-------if (ret < 0)
359 >-------->------->-------goto out;
360 >-------->-------trace_len = ret;
361 >-------->-------ret = 0;
362 #else
363 >-------->-------trace.skip = 0;
364 >-------->-------trace.nr_entries = 0;
365 >-------->-------trace.max_entries = MAX_STACK_ENTRIES;
366 >-------->-------trace.entries = trace_entries;
367 >-------->-------ret = save_stack_trace_tsk_reliable(t, &trace);
368 #endif
369 >-------->-------WARN_ON_ONCE(ret == -ENOSYS);
370 >-------->-------if (ret) {
371 >-------->------->-------pr_info("%s: %s:%d has an unreliable stack\n",
372 >-------->------->------->------- __func__, t->comm, t->pid);
373 >-------->------->-------goto out;
374 >-------->-------}
```

OpenEuler

栈检查



构建检查队列

OpenEuler

指令修改

激活时

```
424 >-------func_node = klp_find_func_node(func->old_func);
425 >-------ip = (unsigned long)func->old_func;
426 >-------if (!func_node) {
427 >-------->-------func_node = func->func_node;
428 >-------->-------if (!func_node)
429 >-------->-------->-------return -ENOMEM;
430
431 >-------->-------INIT_LIST_HEAD(&func_node->func_stack);
432 >-------->-------func_node->old_func = func->old_func;
433 >-------->-------ret = copy_from_kernel_nofault(func_node->old_code,
434 >-------->-------->-------->-------->-------(void *)ip, JMP_E9_INSN_SIZE);
435 >-------->-------if (ret) {
436 >-------->-------->-------return -EPERM;
437 >-------->-------}
438 >-------->-------list_add_rcu(&func_node->node, &klp_func_list);
439 >-------}
```

在热补丁函数队列里寻找是否该函数曾经打过热补丁，若该函数未被打过热补丁，那么备份替换区的指令，并将其加入热补丁函数队列里。

```
441 >-------list_add_rcu(&func->stack_node, &func_node->func_stack);
```

加入该函数的热补丁队列

```
443 >-------new_addr = (unsigned long)func->new_func;
444 >-------/* replace the text with the new text */
445 >-------new = klp_jmp_code(ip, new_addr);
446 >-------text_poke((void *)ip, new, JMP_E9_INSN_SIZE);
```

生成跳转指令，并将原始函数的开头指令替换为此跳转指令

OpenEuler

指令修改

去使能时

```
458 >-------func_node = klp_find_func_node(func->old_func);
459 >-------ip = (unsigned long)func_node->old_func;
460 >-------if (list_is_singular(&func_node->func_stack)) {
461 >------->-------list_del_rcu(&func->stack_node);
462 >------->-------list_del_rcu(&func_node->node);
463 >------->-------new = klp_old_code(func_node->old_code);
464 >-------} else {
465 >------->-------list_del_rcu(&func->stack_node);
466 >------->-------next_func = list_first_or_null_rcu(&func_node->func_stack,
467 >------->------->------->------->------->-------struct klp_func, stack_node);
468
469 >------->-------new_addr = (unsigned long)next_func->new_func;
470 >------->-------new = klp_jmp_code(ip, new_addr);
471 >-------}
```

在热补丁函数队列里找到生效的热补丁，如果该函数只打过一个热补丁，那么获取之前备份的旧指令；如果打过多个热补丁，拿到前一个热补丁的函数地址，并生成跳转指令

```
473 >-------/* replace the text with the new text */
474 >-------text_poke((void *)ip, new, JMP_E9_INSN_SIZE);
```

将原始函数的前几条指令替换为前面拿到的指令。

OpenEuler

Livepatch使用

- 需要打开以下这些config选项
  - CONFIG_HAVE_LIVEPATCH_WO_FTRACE=y
  - CONFIG_LIVEPATCH=y
  - CONFIG_LIVEPATCH_WO_FTRACE=y
  - CONFIG_LIVEPATCH_STOP_MACHINE_CONSISTENCY=y
  - CONFIG_LIVEPATCH_STACK=y
  - CONFIG_LIVEPATCH_RESTRICT_KPROBE=y
  - CONFIG_KALLSYMS=y
  - CONFIG_KALLSYMS_ALL=y
  - CONFIG_DEBUG_INFO=y

Livepatch使用

samples/livepatch/livepatch-sample.c

- 热补丁制作方式
  - 1. 编写模块代码方式

```
67 static struct klp_func funcs[] = {
68 >-------{
69 #ifdef CONFIG_PPC64
70 >-------->-------.old_name = ".cmdline_proc_show",
71 #else
72 >-------->-------.old_name = "cmdline_proc_show",
73 #endif
74 >-------->-------.new_func = livepatch_cmdline_proc_show,
75 >-------->-------.force = 2,
76 >-------}, { }
77 };
78
79 static struct klp_object objs[] = {
80 >-------{
81 >-------->-------/* name being NULL means vmlinux */
82 >-------->-------.funcs = funcs,
83 #ifdef CONFIG_LIVEPATCH_STOP_MACHINE_CONSISTENCY
84 >-------->-------.hooks_load = hooks_load,
85 >-------->-------.hooks_unload = hooks_unload,
86 #endif
87 >-------}, { }
88 };
89
90 static struct klp_patch patch = {
91 >--------.mod = THIS_MODULE,
92 >--------.objs = objs,
93 };
```

```
61 static int livepatch_cmdline_proc_show(struct seq_file *m, void *v)
62 {
63 >-------seq_printf(m, "%s\n", "this has been live patched");
64 >-------return 0;
65 }
```

```
 95 static int livepatch_init(void)
 96 {
 97 #ifdef CONFIG_PPC64
 98 >-------patch.objs[0].funcs[0].new_func =
 99 >-------->-------(void *)ppc_function_entry((void *)livepatch_cmdline_proc_show);
100 #endif
101
102 #ifdef CONFIG_LIVEPATCH_PER_TASK_CONSISTENCY
103 >-------return klp_enable_patch(&patch);
104 #elif defined(CONFIG_LIVEPATCH_STOP_MACHINE_CONSISTENCY)
105 >-------return klp_register_patch(&patch);
106 #endif
107 }
108
109 static void livepatch_exit(void)
110 {
111 #ifdef CONFIG_LIVEPATCH_STOP_MACHINE_CONSISTENCY
112 >-------WARN_ON(klp_unregister_patch(&patch));
113 #endif
114 }
```

OpenEuler

Livepatch使用

- 热补丁制作方式
  - 2. 通过kpatch工具
    - 下载kpatch源码：https://github.com/dynup/kpatch
    - 在kpatch目录下执行make（编译kpatch工具）
    - 准备好一个修改函数内容的patch
    - export NO_PROFILING_CALLS=1（避免报错）
    - 设置对应的ARCH与CROSS_COMPILE
    - 在kpatch-build目录下执行：./kpatch-build -s <src dir> -v <src dir>/vmlinux <patch dir>/xxxx.patch --skip-gcc-check -c <src dir>/.config

Livepatch使用

- 热补丁ko使用
  - 插入内核：insmod xxx.ko
  - 激活热补丁：echo 1 > /sys/kernel/livepatch/xxx/enabled
  - 去使能热补丁：echo 0 > /sys/kernel/livepatch/xxx/enabled
  - 卸载ko：rmmod xxx
  - 查询当前热补丁状态：cat /proc/livepatch/state

  - 注：热补丁激活的顺序需要与插入内核的顺序保持一致，只可去使能最后一个激活的热补丁

实施例

```
diff --git a/kernel/cgroup/cgroup.c b/kernel/cgroup/cgroup.c
index 350297ad63f1..de40f0f33333 100644
--- a/kernel/cgroup/cgroup.c
+++ b/kernel/cgroup/cgroup.c
@@ -6204,6 +6204,7 @@ void cgroup_exit(struct task_struct *tsk)
 >------struct css_set *cset;
 >------int i;
 -
+>------printk("livepatch: out of cgroup\n");
 >------spin_lock_irq(&css_set_lock);
 -
 >------WARN_ON_ONCE(list_empty(&tsk->cg_list));
---
```

① 准备好patch。可用git format-patch来生成：git format-patch -1。
注意：生成patch后记得回退该补丁

```
root@ubuntu1804:/home/zzx/kpatch/kpatch-build# ./kpatch-build -s /home/yeweihua/projects/hulk/hulk-5.10/ -c build/.config -v build/vmlinux --skip-gcc-check /home/yeweihua/projects/hulk/hulk-5.10/0001-test.patch
WARNING: Skipping gcc version matching check (not recommended)
Testing patch file(s)
Reading special section data
Building original kernel
Building patched kernel
sh: 0: Can't open /lkp/bin/lkp_funccheck
Extracting new and modified ELF sections
cgroup.o: changed function: cgroup_exit
Patched objects: vmlinux
Building patch module: livepatch-0001-test.ko
SUCCESS
```

② 使用kpatch工具制作生成livepatch，-s表示源码路径，-c表示.config，-v表示vmlinux

OpenEuler

实施例

```
/modules # insmod livepatch-0001-test.ko
[   18.081533] livepatch_0001_test: loading out-of-tree module taints kernel.
[   18.081995] livepatch_0001_test: tainting kernel with TAINT_LIVEPATCH
[   18.434660] insmod (80) used greatest stack depth: 13952 bytes left
```

③ 将生成的热补丁插入内核

```
/modules # echo 1 > /sys/kernel/livepatch/livepatch_0001_test/enabled
[  311.169099] livepatch: tainting kernel with TAINT_LIVEPATCH
[  311.169099] livepatch: enabling patch 'livepatch_0001_test'
```

④ 将热补丁激活

```
/modules # ls
[  382.479847] livepatch: out of cgroup
livepatch-0001-test.ko  livepatch-sample.ko     livepatch-sample2.ko
```

⑤ 验证热补丁是否生效

```
/modules # echo 0 > /sys/kernel/livepatch/livepatch_0001_test/enabled
[  704.205115] livepatch: disabling patch 'livepatch_0001_test'
```

⑥ 去使能热补丁

```
/modules # rmmod livepatch-0001-test.ko
```

⑦ 卸载热补丁

```
/modules # cat /proc/livepatch/state
Index   Patch                                   State
----------------------------------------------------
1       livepatch_0001_test                     disabled
----------------------------------------------------
```

查询当前系统的热补丁状况

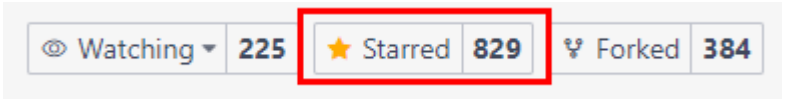
OpenEuler

✓ openEuler kernel gitee 仓库

源代码仓库

https://gitee.com/openeuler/kernel

欢迎大家多多 Star，多多参与社区开发，多多贡献补丁。



✓ maillist、issue、bugzilla

可以通过邮件列表、issue、bugzilla 参与社区讨论

欢迎大家多多讨论问题，发现问题多提 issue、bugzilla

https://gitee.com/openeuler/kernel/issues

https://bugzilla.openeuler.org

kernel@openeuler.org

✓ openEuler kernel SIG 微信技术交流群

请扫描右方二维码添加小助手微信

或者直接添加小助手微信（微信号：openeuler-kernel）

备注“交流群”或“技术交流”

加入 openEuler kernel SIG 技术交流群

技术交流

OpenEuler

Thank you

OpenEuler