## 文献阅读

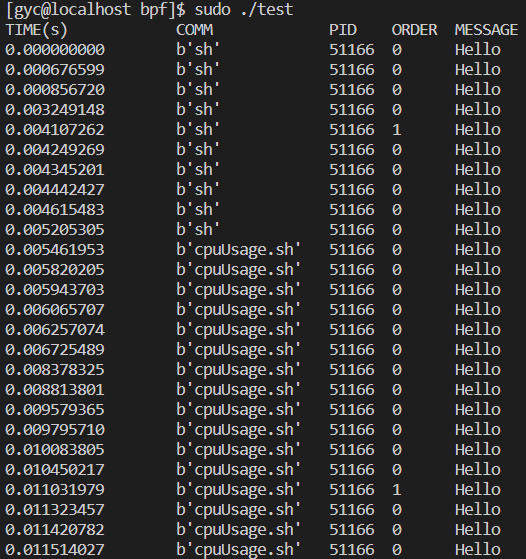
阅读了《Linux内核观测技术BPF》，主要学习了具体的BPF程序类型、BPF 映射类型和使用方法等；

正在学习BCC开发文档：[bcc/reference\_guide.md at master · iovisor/bcc (github.com)](https://github.com/iovisor/bcc/blob/master/docs/reference_guide.md" \l "initialization)；

以及一些相关的博客；

## BCC

因为是使用python编写（其实官方更推荐go，但是我没学过），而且高度集成，所以先使用BCC开发一下来构建一个大概的框架，后面使用纯C或者CO-RE开发起来也方便，目前只是测试了能够追踪具体调用：



alloc\_pages 可以使用 kprobe探针进行跟踪，主要是使用了BCC中的 bpf\_get\_current\_pid\_tgid()、bpf\_get\_current\_comm() 等获取调用信息，然后用 Perf 事件数组映射 来实时向用户发送。

接下来主要任务是看完BCC文档，然后使用BPF HASH MAP来对pid进行映射，从而统计进程的调用信息。其次就是对函数参数中的gfp\_mask也解析一下，然后尝试获取函数调用栈（这个还没研究明白）。

对于程序的输出打算实现两种格式：

单进程，需要指定pid，然后就串行输出每一次的调用信息，包括order、mask、调用栈等；

全部进程，就根据order来统计获取的页数，并且根据最近最多调用的排序来实现动态显示。

还有一个问题就是，这个使用BCC开发并完善以后可以并入到bcc库中，作为一个bcc-tool，而不是kernel。要提交到kernel的话我只能想到在内核的samples/bpf下添加新的样例。

## libbpf-bootstrap

（只是看了博客和相关介绍，没有实践过：[libbpf/libbpf-bootstrap：使用 libbpf 和 BPF CO-RE 进行 BPF 应用程序开发的脚手架 (github.com)](https://github.com/libbpf/libbpf-bootstrap)）

这个是git上的一个开源项目，是使用纯C编写bpf程序比较方便的做法，libbpf-bootstrap是基于libbpf开发BPF程序的简易开发框架，它依赖于libbpf、bpftool，并且将它们以git submodule的形式配置到其项目中。

在配置好环境之后，使用这个框架开发一个新的程序比较简单，libbpf\_bootstrap项目做了一个很有扩展性的Makefile，我们只需在特定文件夹下分别编写好内核和用户文件，然后在Makefile中的APP变量后面增加一个条目即可。

与之对应的，之前我也尝试过直接在samples/bpf编写程序并修改Makefile，感觉这个实际上就是一个简化版的过程，可以不下载内核代码，原理上一样。

## CO-RE

后续我又看了一些资料，这个相关原理和开发过程比我想得要的复杂一些，因此打算先完成上面两个之后再进行可移植的开发。

BPF CO-RE 的目标是：

* 作为一种简单的方式帮助 BPF 开发者解决简单的移植性问题（例如读取结构体的字段）
* 并且作为一种仍然可行（不是最优，但可容忍）的方式 解决复杂的移植性问题（例如不兼容的数据结构改动、复杂的用户空间控制条件等）
* 使得开发者能遵循”一次编译、到处运行“（Compile Once – Run Everywhere）范式。

这是通过几个 BPF CO-RE 模块的组合实现的：

* vmlinux.h 消除了对内核头文件的依赖；
* 字段重定位信息（字段偏置、字段是否存在、字段大小等等）使得从内核提取数据这个过程变得可移植；
* libbpf 提供的 Kconfig extern 变量允许 BPF 程序适应不同的内核版本 —— 以及配置相关的差异；
* 当其他方式都失效时，应用提供的只读配置和 struct flavor 最终救场，能解决 任何需要复杂处理的场景。

相关资料有：

[[译] BPF 可移植性和 CO-RE（一次编译，到处运行）【转载】 | 深入浅出 eBPF](https://www.ebpf.top/post/bpf_core/)

[BPF CO-RE 参考指南 (nakryiko.com)](https://nakryiko.com/posts/bpf-core-reference-guide/)

[bpf\_study/index.md at master · DavadDi/bpf\_study (github.com)](https://github.com/DavadDi/bpf_study/blob/master/bpf-co-re-btf-libbpf/index.md)

[【译】BTFGen: 让 eBPF 程序可移植发布更近一步 | 深入浅出 eBPF](https://www.ebpf.top/post/btfgen-one-step-closer-to-truly-portable-ebpf-programs/)