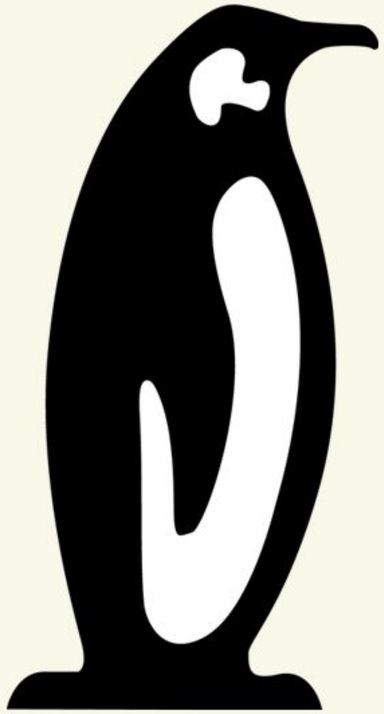


Linux Plumbers Conference

Richmond, Virginia | November 13-15, 2023

Linux Plumbers Conference | Richmond, VA | Nov. 13-15, 2023

Simplified Android Kernel Driver Development with DDK v2

Matthias Männich <maennich@android.com> | Android Microconference

DDK - Overview

Android 14 - DDK definition embedded in source tree

```c
// my_mod.c

#include <linux/module.h>



MODULE_DESCRIPTION("A demo module");
MODULE_LICENSE("GPL v2");

void print_from_my_mod(void) {
  printk(KERN_INFO "Hello");
};

EXPORT_SYMBOL_GPL(print_from_my_mod);
```

```c
// my_other_mod.c

#include <linux/module.h>

#include "my_mod.h"

MODULE_DESCRIPTION("Another demo module");
MODULE_LICENSE("GPL v2");

void print_something(void) {
    print_from_my_mod();
};
```

```bazel
# BUILD.bazel

load("//build/kernel/kleaf:kernel.bzl", "ddk_module")

ddk_module(
    name = "my_mod",
    srcs = ["my_mod.c",],
    out = "my_mod.ko",
    hdrs = ["my_mod.h"],
    kernel_build = "//common:kernel",
    deps = ["//common:all_headers"],
)

ddk_module(
    name = "my_other_mod",
    srcs = ["my_other_mod.c",],
    out = "my_other_mod.ko",
    kernel_build = "//common:kernel",
    deps = [
        ":my_mod",
        "//common:all_headers",
    ],
)
```

DDK - Overview

Android 14

- Consistent toolchain selection (clang, hermetic toolchain)
- Subset of kernel sources visible to modules (some headers!)
- Build file generation (Kbuild, Makefile, Kconfig)
- Dependency resolution
- Build orchestration (via Kleaf using Bazel)
- Packaging / Image generation

Documentation: https://android.googlesource.com/kernel/build/+/refs/heads/main/kleaf/docs/ddk/main.md

Android 15 - NEW! - "From empty directory to flashable kernel module in less than 5 minutes" *

- Build against prebuilt (signed) GKI images that are
  - on your disk
  - hosted on ci.android.com
  - hosted privately

- Prebuilts come with batteries included
  - hermetic toolchain (compilers, sysroot, etc.)
  - archives of headers

* subject to bandwidth and local compute resources

```
# WORKSPACE

kleaf_repository(
    name = "kleaf",
    local_repo_dir = "path/to/kleaf/tooling/repo",
    prebuilt_artifact_dir = "path/to/distributed/prebuilts",
)
```

```
# BUILD.bazel

load("@kleaf//build/kernel/kleaf:kernel.bzl", "ddk_module")

ddk_module(
    name = "my_mod",
    srcs = ["my_mod.c",],
    out = "my_mod.ko",
    kernel_build = "@gki_prebuilts//:kernel_aarch64",
)
```

Android 15 - NEW! - Build against

- a fixed build on ci.android.com

```
kleaf_repository(
    name = "kleaf",
    build_number = "123456789",
)
```

- a released tag

```
kleaf_repository(
    name = "kleaf",
    branch = "android15-6.1-2024-01_r1",
)
```

- moving targets

```
kleaf_repository(
    name = "kleaf",
    // slowly moving release branch
    branch = "android15-6.1-2024-01",
)

kleaf_repository(
    name = "kleaf",
    // development branch
    branch = "android15-6.1",
)

kleaf_repository(
    name = "kleaf",
    // mainline linux + Android patches
    branch = "android-mainline",
)
```

Android 15 - NEW! - Upgrading kernels could be as simple as a one-line change

- Upgrade within same LTS version

```
  kleaf_repository(
      name = "kleaf",
-      branch = "android14-6.1",
+      branch = "android15-6.1",
  )
```

- Upgrade to next LTS version

```
  kleaf_repository(
      name = "kleaf",
-      branch = "android15-6.1",
+      branch = "android15-6.6",
  )
```

- Upgrade to more recent release branch

```
  kleaf_repository(
      name = "kleaf",
-      branch = "android15-6.1-2023-03",
+      branch = "android15-6.1-2023-10",
  )
```

Of course, compile/link/test issues are not fixed yet.

Android 15 - NEW! - Upgrading kernels could be as simple as a one-line change

- DDK helps transitioning to newer kernel versions

```
// my_mod.c

#include <linux/module.h>

MODULE_DESCRIPTION("A demo module");
MODULE_LICENSE("GPL v2");

void print_from_my_mod(void) {
#ifdef DDK_ANDROID_14
  printk(KERN_INFO "Hello from Android 14");
#elif DDK_ANDROID_15
  printk(KERN_INFO "Hello from Android 15");
#else
  printk(KERN_INFO "Hello");
#endif
};
```

Work-in-progress API for illustration purposes.

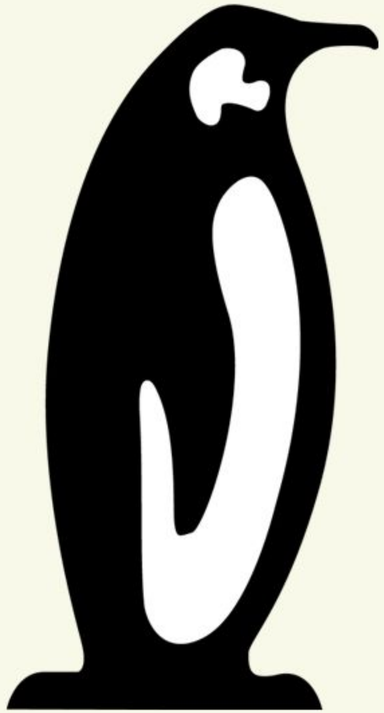
Android 15 - NEW! - What about upstreaming DDK modules?

YES, please!

- DDK produces upstream-friendly Kbuild files
- DDK provides a framework to encourage upstream-friendly module development
- Dependency unwinding helps deciding on the upstreaming order
- Modules can be built conditionally based on the kernel built against
  - e.g. my_mod should only be built for 6.1 kernels as 6.2+ contains the module upstream

Questions?

Matthias Männich <maennich@android.com> | Android Microconference



Linux Plumbers Conference

Richmond, Virginia | November 13-15, 2023