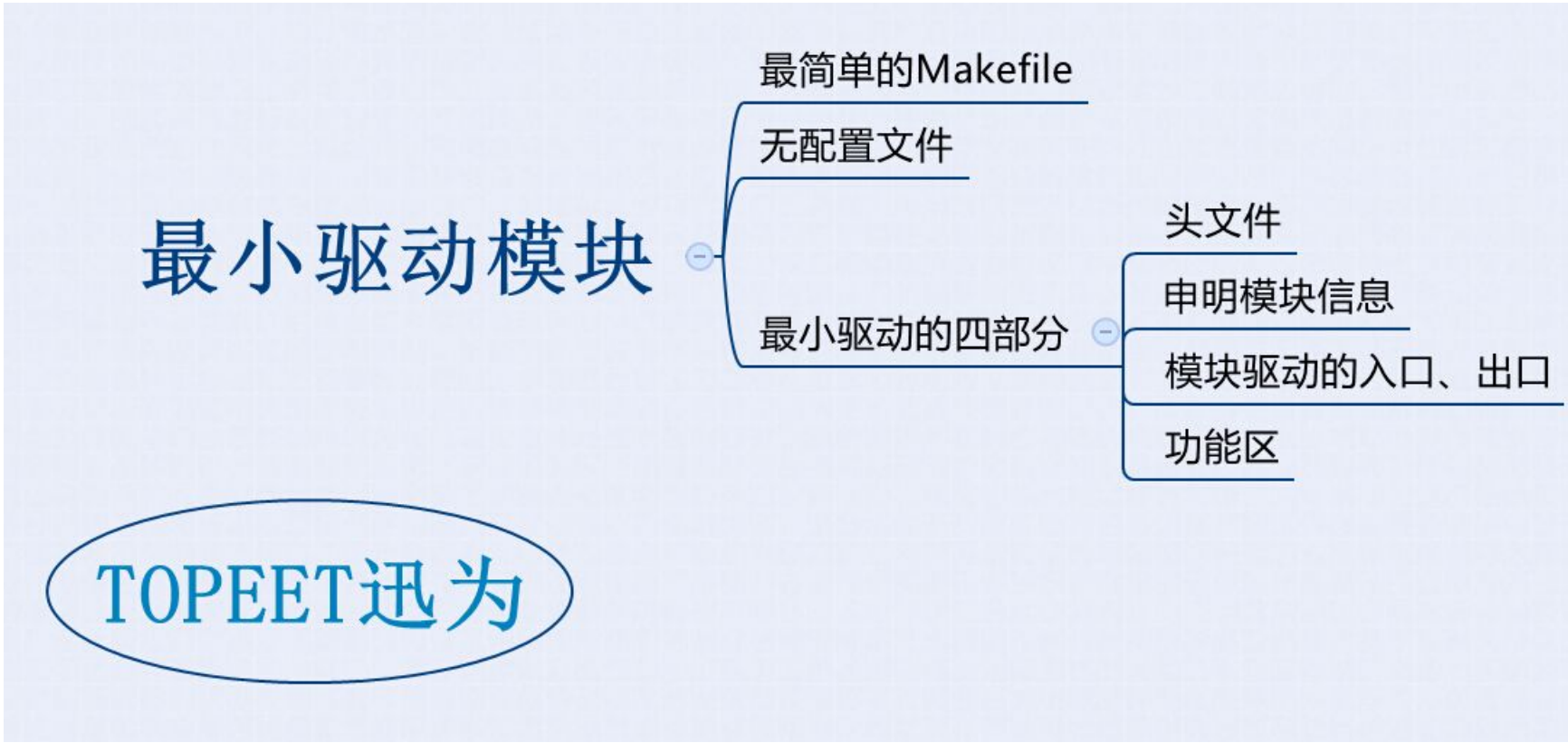
迅为电子Linux驱动教程
——DriverModule_01

北京迅为电子有限公司

主要内容

- 最简Linux驱动
  - 完成的功能是：加载驱动的时候打印“helloxx”
- 模块的Makefile文件
  - 驱动模块的编译分析
- 编译、加载、卸载模块（开发板实际操作）

最简Linux驱动


最小驱动模块
最简单的Makefile
无配置文件
最小驱动的四部分
头文件
申明模块信息
模块驱动的入口、出口
功能区
TOPEET迅为

最简Linux驱动
——必备的头文件

- Linux头文件位置
  - 类似#include <linux/module.h>的头文件，它们是在Linux源码目录下的include/linux/module.h

- #include <linux/module.h>头文件
  - 所有的Linux 代码必须遵循GPL 协议，如果不知道Linux 的GPL 协议，去查一下资料
  - 如果你不声明GPL 协议，你的模块将无法在Linux 中使用的
  - MODULE_LICENSE(_license)**添加遵循GPL协议，必须的！**
  - MODULE_AUTHOR(_author)代码作者

最简Linux驱动
——必备的头文件

- #include <linux/init.h>
  - 包含初始化宏定义的头文件,代码中的函数module_init和module_exit在此文件中
  - 入口函数module_init(x)
  - 出口函数module_exit(x)

- 新建.c文件
  - mini_linux_module.c

最简Linux驱动
——模块的入口和出口

- module_init(hello_init);
  - /*初始化函数*/

- module_exit(hello_exit);
  - /*卸载函数*/

最简Linux驱动
——声明区

- 声明区
  - MODULE_LICENSE("Dual BSD/GPL");
  - /*声明是开源的，没有内核版本限制*/
  - **必须有**

  - MODULE_AUTHOR("TOPEET");
  - /*声明作者*/
  - 可有可无

最简Linux驱动
——功能区

```
static int hello_init(void){
        printk(KERN_EMERG "Hello World enter!\n");
        /*打印信息，KERN_EMERG表示紧急信息*/
        return 0;
}
static void hello_exit(void)
{
        printk(KERN_EMERG "Hello world exit!\n");
}
```

最简Linux驱动

```c
#include <linux/init.h>
/*包含初始化宏定义的头文件,代码中的module_init和module_exit在此文件中*/
#include <linux/module.h>
/*包含初始化加载模块的头文件,代码中的MODULE_LICENSE在此头文件中*/


MODULE_LICENSE("Dual BSD/GPL");
/*声明是开源的，没有内核版本限制*/
MODULE_AUTHOR("TOPEET");
/*声明作者*/


static int hello_init(void)
{
    printk(KERN_EMERG "Hello World enter!\n");
    /*打印信息，KERN_EMERG表示紧急信息*/
    return 0;
}

static void hello_exit(void)
{
    printk(KERN_EMERG "Hello world exit!\n");
}


module_init(hello_init);
/*初始化函数*/
module_exit(hello_exit);
/*卸载函数*/
```

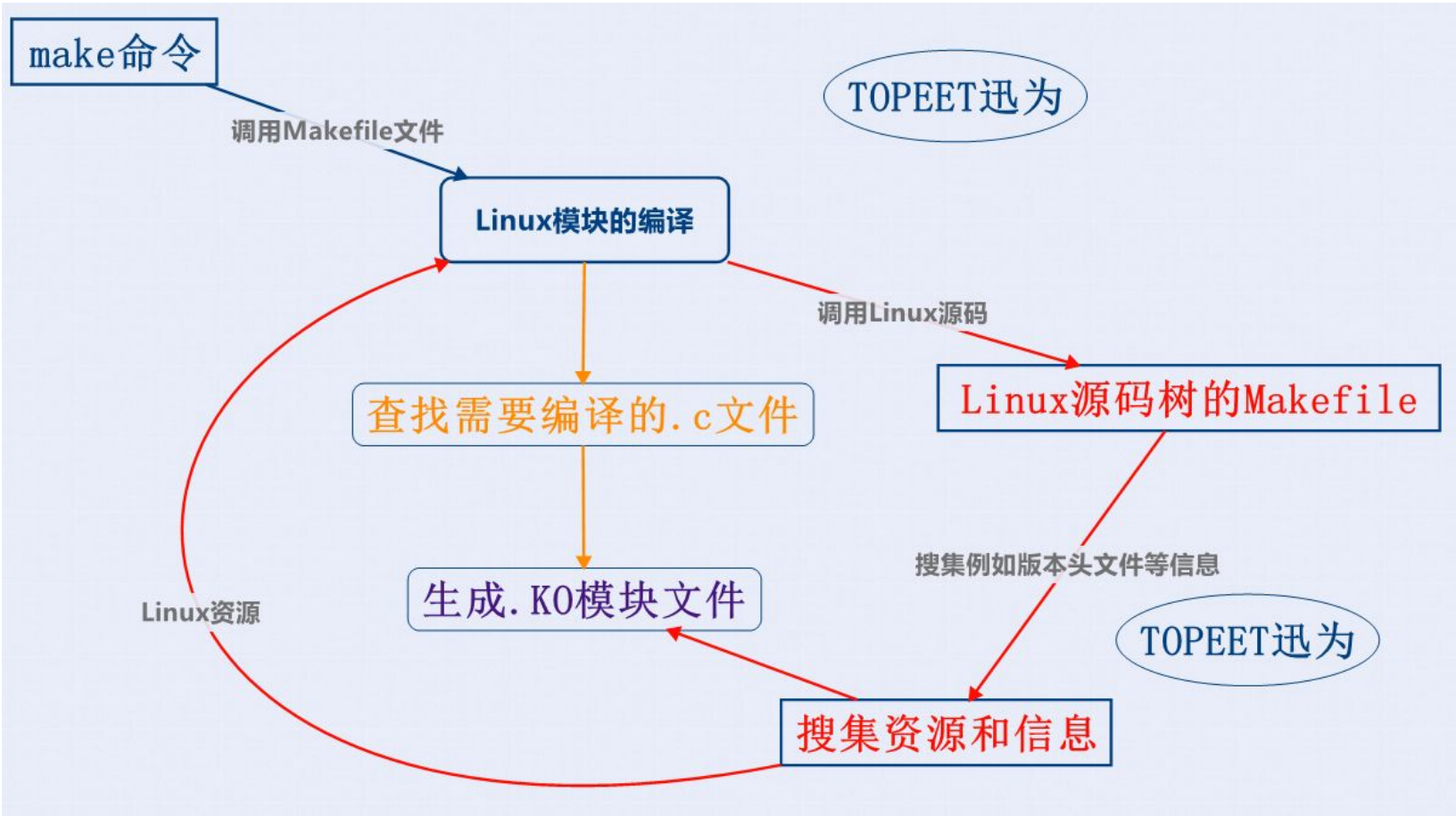
驱动模块的编译

- Linux的驱动可以和Linux源码放在一起编译，也可以单独拿出来编译。
  - 为了理解整个Linux内核编译过程，可以从学习Linux模块的编译入手
- Makefile文件
  - 单独编译驱动需要写一个Makefile文件
- 编写Makefile文件的最好方式是“依葫芦画瓢”
  - 以后可能会碰到各种脚本，脚本的语法是学不完的

驱动模块的编译

```
#!/bin/bash
#通知编译器我们要编译模块的哪些源码
#这里是编译itop4412_hello.c这个文件编译成中间文件mini_linux_module.o
obj-m += mini_linux_module.o

#源码目录变量，这里用户需要根据实际情况选择路径
#作者是将Linux的源码拷贝到目录/home/topeet/android4.0下并解压的
KDIR := /home/topeet/android4.0/iTop4412_Kernel_3.0

#当前目录变量
PWD ?= $(shell pwd)

#make命名默认寻找第一个目标
#make -C就是指调用执行的路径
#$(KDIR)Linux源码目录，作者这里指的是/home/topeet/android4.0/iTop4412_Kernel_3.0
#$(PWD)当前目录变量
#modules要执行的操作
all:
	make -C $(KDIR) M=$(PWD) modules

#make clean执行的操作是删除后缀为o的文件
clean:
	rm -rf *.o

```

编译流程分析


make命令
调用Makefile文件
TOPEET迅为
Linux模块的编译
调用Linux源码
查找需要编译的.c文件
Linux源码树的Makefile
搜集例如版本头文件等信息
生成.KO模块文件
Linux资源
TOPEET迅为
搜集资源和信息

实验操作

——编译文件

- 拷贝两个文件到虚拟机，执行Make命令，编译生成KO文件
  - 在window下写的Makefile文件拷贝到Linux中可能会有点小问题需要修改
  - 主要是Tab问题
  - all和clean参数后面的必须添加Tab键，否则会报错“*** missing separator. Stop”

- 编译会生成KO文件，KO就是驱动模块

实验操作
——加载模块和卸载模块

- 开发板运行最小Linux系统
- 使用U盘（或者TF卡），将KO文件拷贝到Linux最小系统
  - 最小系统给大家提供，直接烧写即可
- 加载U盘
  - 参考使用手册
- 加载模块、查看模块、卸载模块
  - insmod加载模块命令
  - lsmod查看模块命令
  - rmmod卸载模块命令

小结

- linux代码中记得添加GPL协议
- 掌握驱动入口和出口函数的调用（也可以说是宏的调用）
- 对Makefile文件有简单的了解，会仿写
- 掌握加载、查看、卸载模块的命令

谢谢！