实验1 PolyOS AIoT安装和内核编译与安装

郝淼 202328013229045

任务一：构建、运行 PolyOS

构建 PolyOS：

```
git clone https://gitee.com/riscv-raios/build_portal.git && cd build_portal
PATH=${PATH}:~/.local/bin kas build common-oscourse-qemuriscv64.yml
```

构建结束后，命令行输出如图：

```
NOTE: Tasks Summary: Attempted 3487 tasks of which 0 didn't need to be rerun and all succeeded.
NOTE: Writing buildhistory
NOTE: Writing buildhistory took: 2 seconds
NOTE: Build completion summary:
NOTE:   do_populate_sysroot: 0.0% sstate reuse(0 setscene, 236 scratch)
NOTE:   do_deploy_source_date_epoch: 0.0% sstate reuse(0 setscene, 287 scratch)
NOTE:   do_package_qa: 0.0% sstate reuse(0 setscene, 130 scratch)
NOTE:   do_package: 0.0% sstate reuse(0 setscene, 130 scratch)
NOTE:   do_packagedata: 0.0% sstate reuse(0 setscene, 130 scratch)
NOTE:   do_package_write_ipk: 0.0% sstate reuse(0 setscene, 130 scratch)
NOTE:   do_populate_lic: 0.0% sstate reuse(0 setscene, 287 scratch)
```

在 QEMU 上运行 PolyOS：

```
kas shell common-qemuriscv64-core-image-minimal.yml -c 'runqemu nographic'
```

BIOS 启动：

```
OpenSBI v0.9
   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|

Platform Name             : riscv-virtio,qemu
Platform Features         : timer,mfdeleg
Platform HART Count       : 4
Firmware Base             : 0x80000000
Firmware Size             : 124 KB
Runtime SBI Version       : 0.2
```

用户登录：

```
PolyOS PolyOS.1.0 qemuriscv64 ttyS0

qemuriscv64 login: root
root@qemuriscv64:~#
```

常用命令：

```
root@qemuriscv64:~# uname -r
5.15.6-yocto-standard
root@qemuriscv64:~# uname -a
Linux qemuriscv64 5.15.6-yocto-standard #1 SMP PREEMPT Wed Sep 27 08:34:55 UTC 2023 riscv64 GNU/
root@qemuriscv64:~# date
Fri Nov 19 17:23:56 UTC 2021
root@qemuriscv64:~#
```

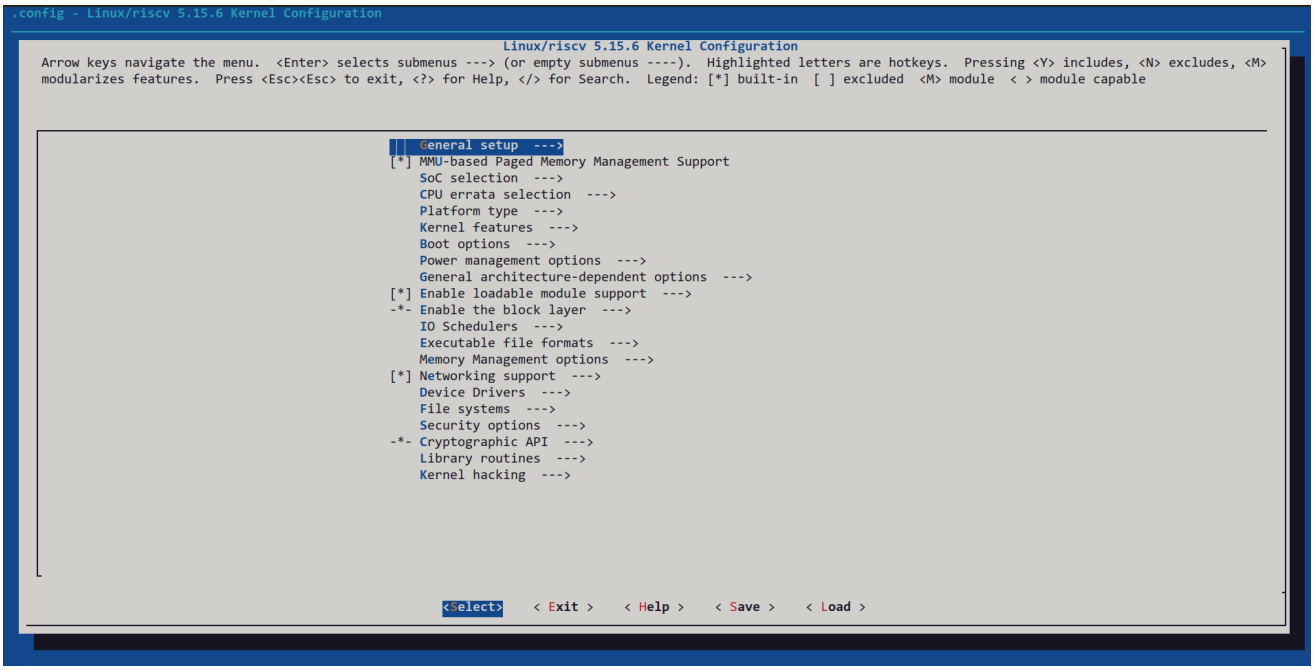
输入 poweroff 关机。

任务二：内核编译与安装

打开 Kconfig 的命令行图形配置界面：

```
kas shell common-oscourse-qemuriscv64.yml -c 'bitbake linux-yocto -c menuconfig'
```

效果如图：



配置完成后，通过 Exit 选项退出 menuconfig，然后编译并安装内核：

```
kas shell common-oscourse-qemuriscv64.yml -c 'bitbake linux-yocto'
```

编译安装内核结束后，输出如下：

```
NOTE: Executing Tasks
NOTE: Tasks Summary: Attempted 664 tasks of which 654 didn't need to be rerun and all succeeded.
NOTE: Writing buildhistory
NOTE: Writing buildhistory took: 0 seconds
NOTE: Build completion summary:
NOTE:   do_populate_sysroot: 0.0% sstate reuse(0 setscene, 1 scratch)
NOTE:   do_package_qa: 0.0% sstate reuse(0 setscene, 1 scratch)
NOTE:   do_package: 0.0% sstate reuse(0 setscene, 1 scratch)
NOTE:   do_packagedata: 0.0% sstate reuse(0 setscene, 1 scratch)
NOTE:   do_package_write_ipk: 0.0% sstate reuse(0 setscene, 1 scratch)
```

任务三：内核模块编程

在 build_portal/meta-oscourse/recipes-modules 下新建以学号命名的目录项：

```
.
├── 202328013229045
├── hello
├── kmalloc
├── request_mem_region
├── request_region
└── vmalloc
```

在 202328013229045 下创建若干文件：

```
202328013229045/
├── 202328013229045.bb
└── files
    ├── 202328013229045.c
    └── Makefile
```

各文件内容：

```
$ cat 202328013229045.bb
LICENSE = "CLOSED"

SRC_URI = "file://202328013229045.c \
           file://Makefile \
         "

inherit module

S = "${WORKDIR}"

# The inherit of module.bbclass will automatically name module packages with
# "kernel-module-" prefix as required by the oe-core build environment.

RPROVIDES:${PN} += "kernel-module-202328013229045"

$ cat files/202328013229045.c
#include <linux/module.h>

static int __init my_module_init(void)
{
        printk(KERN_ALERT"hello world!\n");
```

```
        return 0;
}

static void __exit my_module_exit(void)
{
}

module_init(my_module_init);
module_exit(my_module_exit);
MODULE_LICENSE("GPL");
MODULE_AUTHOR("Miao Hao");

$ cat files/Makefile
obj-m += 202328013229045.o

SRC := $(shell pwd)

all:
        $(MAKE) -C $(KERNEL_SRC) M=$(SRC)

modules_install:
        $(MAKE) -C $(KERNEL_SRC) M=$(SRC) modules_install

clean:
        $(MAKE) -C $(KERNEL_SRC) M=$(SRC) clean
```

修改 build_portal/meta-oscourse/conf/layer.conf:

```
14,15c14,15
< MACHINE_ESSENTIAL_EXTRA_RRECOMMENDS += "hello kmalloc-mod vmalloc-mod request-region request-mem-region "
< KERNEL_MODULE_AUTOLOAD += "hello kmalloc-mod vmalloc-mod request-region request-mem-region "
---
> MACHINE_ESSENTIAL_EXTRA_RRECOMMENDS += "hello kmalloc-mod vmalloc-mod request-region request-mem-region 202328013229045 "
> KERNEL_MODULE_AUTOLOAD += "hello kmalloc-mod vmalloc-mod request-region request-mem-region 202328013229045 "
```

编译自定义模块:

```
kas shell common-oscourse-qemuriscv64.yml -c 'bitbake 202328013229045'
```

重新构建 PolyOS，并在 QEMU 中运行:

```
kas build common-oscourse-qemuriscv64.yml
kas shell common-qemuriscv64-core-image-minimal.yml -c 'runqemu nographic'
```

登录后，内核已自动加载自定义模块，并可通过 dmesg 查看加载输出:

```
PolyOS PolyOS.1.0 qemuriscv64 ttyS0

qemuriscv64 login: root
root@qemuriscv64:~# lsmod
Module                  Size  Used by
sch_fq_codel           24576  1
hello                  16384  0
202328013229045        16384  0
root@qemuriscv64:~# dmesg | grep hello
[    4.001208] hello world!
[    4.036383] hello world!
root@qemuriscv64:~#
```

卸载自定义模块，然后重新加载，并查看输出：

```
root@qemuriscv64:~# rmmod 202338013229045
rmmod: ERROR: Module 202338013229045 is not currently loaded
root@qemuriscv64:~# rmmod 202328013229045
root@qemuriscv64:~# lsmod
Module                  Size  Used by
sch_fq_codel           24576  1
hello                  16384  0
root@qemuriscv64:~# insmod /lib/modules/5.15.6-yocto-standard/extra/202328013229045.ko
[   59.096923] hello world!
root@qemuriscv64:~# lsmod
Module                  Size  Used by
202328013229045        16384  0
sch_fq_codel           24576  1
hello                  16384  0
root@qemuriscv64:~# dmesg | grep hello
[    3.750615] hello world!
[    3.784415] hello world!
[   59.096923] hello world!
root@qemuriscv64:~#
```