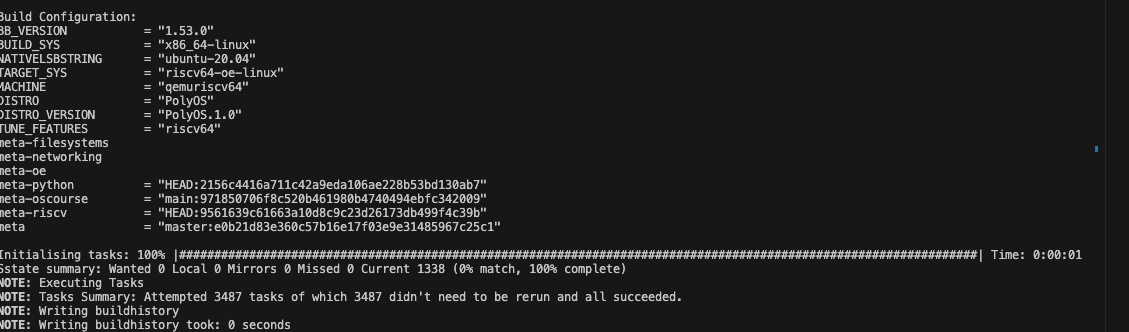
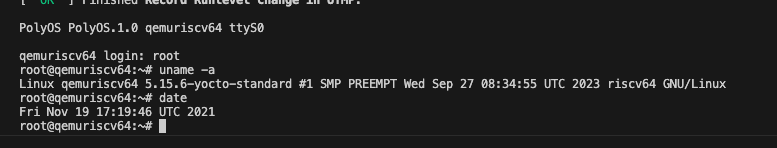
1. 系统环境配置
2. PolyOS的编译：

PATH=${PATH}:~/.local/bin kas build common-oscourse-qemuriscv64.yml



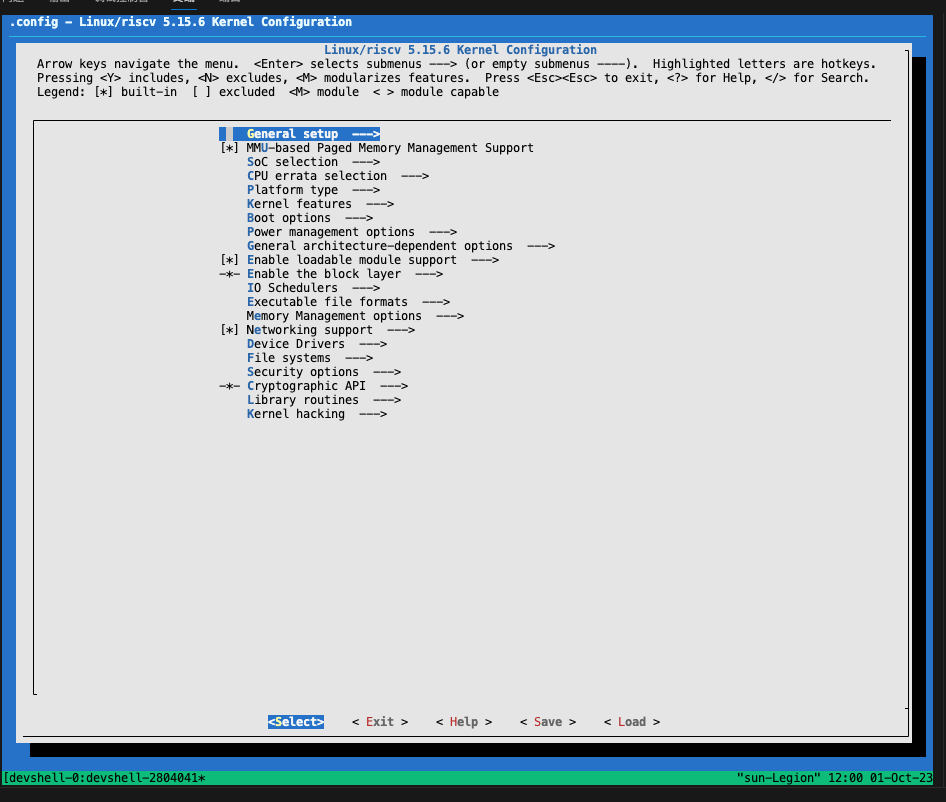
（2）运行PolyOS AIoT：

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



（3）安装基于tumx的图形化编译配置工具：

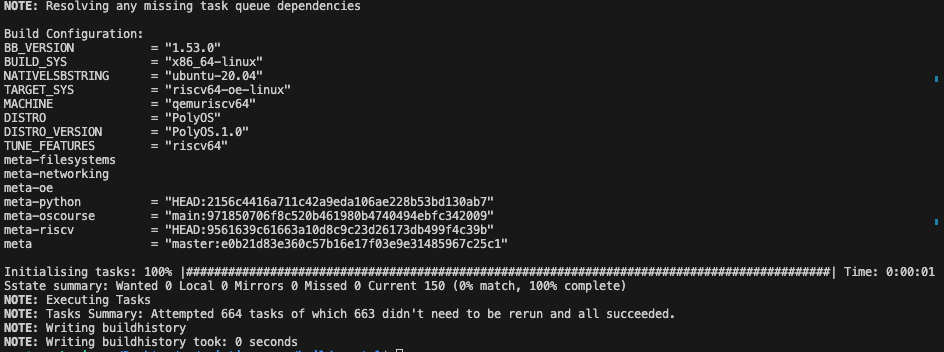
kas shell common-oscourse-qemuriscv64.yml -c 'bitbake linux-yocto -c menuconfig'



（4）内核hello模块的编译与安装：

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

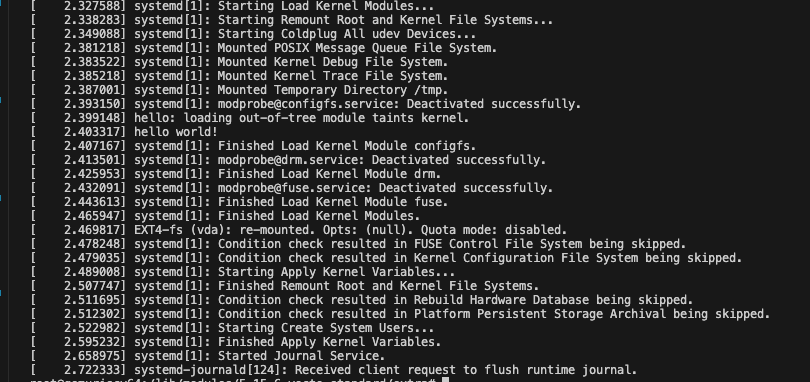
kas shell common-oscourse-qemuriscv64.yml -c 'bitbake hello'



（5）重新编译内核：

kas build common-oscourse-qemuriscv64.yml

（6）dmesg查看printk打印的字符串内容



1. 自定义内核模块修改

在/home/sun/Desktop/os/existing\_repo/build\_portal/meta-oscourse/recipes-modules目录下的hello模块：

1. 修改printk中打印的字符串“hello swh os work!”
2. 修改.c文件名、模块目录名为学号（2023e8013282127），并在Makefile入口、配方文件SRC\_URI与RPROVIDES:${PN}对“hello”做相应替换



1. 在/home/sun/Desktop/os/existing\_repo/build\_portal/meta-oscourse/conf目录下修改layer.conf文件中的MACHINE\_ESSENTIAL\_EXTRA\_RRECOMMENDS、KERNEL\_MODULE\_AUTOLOAD两项包含的模块名称为学号
2. 运行指令，更新模块编译内容：

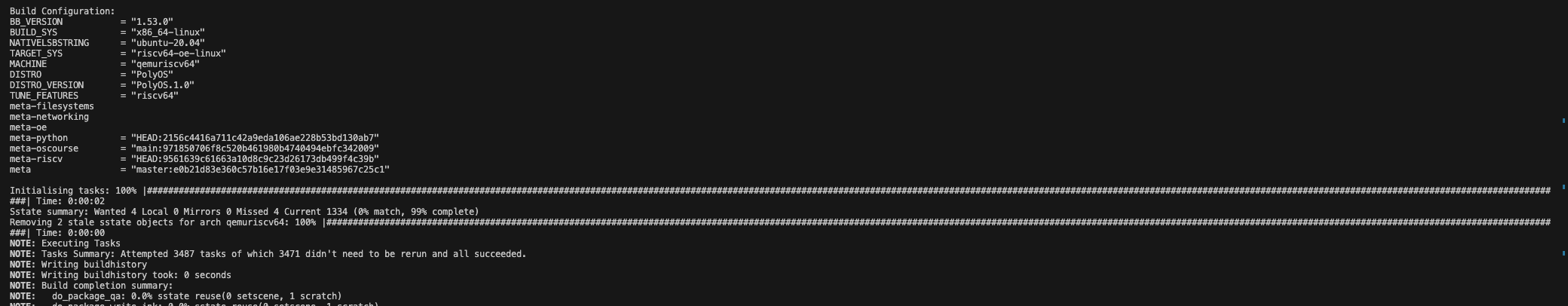
kas shell common-oscourse-qemuriscv64.yml -c 'bitbake -f -c do\_clean 2023e8013282127'

1. 运行指令，编译自定义模块：

kas shell common-oscourse-qemuriscv64.yml -c 'bitbake 2023e8013282127'

1. 运行指令，编译内核：

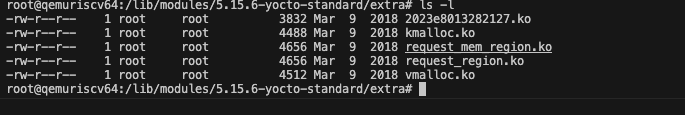
kas build common-oscourse-qemuriscv64.yml，



1. 自定义模块查看
2. 进入qemu:

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

1. 进入/lib/modules/5.15.6-yocto-standard/extra目录下查看新增加的ko文件



1. 查看自定义模块被成功加载：

lsmod | grep 2023e



（4）dmesg查看内核信息，如图，可打印自定义模块中的字符串。

