1. Creating Kernel Modules

- 按步骤来，记录一下遇到的环境配置问题

```
➜  ch2 sudo make
make -C /lib/modules/5.14.16-arch1-1/build M= modules
make[1]: Entering directory '/usr/lib/modules/5.14.16-arch1-1/build'
scripts/Makefile.build:44: arch/x86/entry/syscalls/Makefile: No such file or directory
make[2]: *** No rule to make target 'arch/x86/entry/syscalls/Makefile'.  Stop.
make[1]: *** [arch/x86/Makefile:231: archheaders] Error 2
make[1]: Leaving directory '/usr/lib/modules/5.14.16-arch1-1/build'
make: *** [Makefile:4: all] Error 2
```

- 善用搜索，改 $(PWD) 为 $(shell pwd) 即可:
- `https://stackoverflow.com/questions/39107811/no-rule-to-make-target-arch-x86-entry-syscalls-syscall-32-tbl-needed-by-arch`
- 环境变量中无 PWD
- module 的加载：

```
[  294.047923] simple2: loading out-of-tree module taints kernel.
[  294.048005] simple2: module verification failed: signature and/or required key missing - tainting kernel
[  294.048415] Loading Module
[  294.061020] audit: type=1106 audit(1636441394.073:93): pid=2706 uid=1000 auid=1000 ses=1 msg='op=PAM:sess
```

```
[  424.719280] audit: type=1105 audit(1636441524.731:
ess'
[  424.739079] Removing Module
[  424.745906] audit: type=1106 audit(1636441524.764:
cess'
```

2. Kernel Data Structures

- 做这个时感觉很模糊，网上看了一篇文章，恍然大悟:
- `https://www.cnblogs.com/yangguang-it/p/11667772.html`
- 记住函数中的那些结构体指针都只是一个 “记号”，在函数中都会具体赋值。
- 认识 offsetof 与 container_of 宏的伟大之处
- 简单的总结：
  `https://github.com/liukanglai/Learing/blob/master/Computer/OS/list.md`
- 最好的办法还是看 list.h 的原代码，水平还不够...

```
ess'
[  555.088574] Loading Module
[  555.088577] day: 2, month: 8, year: 1995
[  555.088579] day: 3, month: 9, year: 1996
[  555.088580] day: 4, month: 10, year: 1997
[  555.088582] day: 5, month: 11, year: 1998
[  555.088592] day: 6, month: 12, year: 1999
[  555.091833] audit: type=1106 audit(1636441655
cess'
```

3. UNIX Shell and History Feature

- 程序还是不复杂，四个函数：输入，执行，存历史，输出历史。
- 具体实现真麻烦啊，好多细节都要 debug，但好歹也能 debug，像上面那个就只能反复试错，有时还不得不重启，
- 关于读入的函数，参考
  `http://www.csl.mtu.edu/cs4411.ck/www/NOTES/process/fork/shell.c`
- 写的真的好，不过改了 gets() 为 fgets()，一个回车引出好多问题，果然细节重要

- 加了后台 '&' 的实现，注意到对 args 赋 NULL，才会结束执行，而不是内容为空
- 存历史的时候开了10个数组，注意满和未满的情况，还有开始与结束的下标处理
- 写代码时遇到好多问题，现在回过头来看，也没多少可写，果然还是得“跳出来”

```
osh>who
liukanglai pts/0        2021-11-07 21:56 (tmux(422).%0)
liukanglai tty1         2021-11-07 21:56 (:0)
liukanglai pts/1        2021-11-07 21:57 (:0)
liukanglai pts/2        2021-11-07 21:57 (:0)
liukanglai pts/3        2021-11-07 21:57 (tmux(422).%1)
osh>history
10 who
9 ls
8 ls
7 cal
6 ps
5 ls -l
4 pwd
3 cal
2 who
1 date
osh>!4
4 pwd
/home/liukanglai/Learing/Code/CLionProjects/democ/OS/final-src-osc10e/ch3
osh>history
10 pwd
9 who
8 ls
7 ls
6 cal
5 ps
4 ls -l
3 pwd
2 cal
1 who
osh>!!
10 pwd
/home/liukanglai/Learing/Code/CLionProjects/democ/OS/final-src-osc10e/ch3
osh>ls &
osh>1.txt  DateClient.java      fig3-30.c  fig3-32.c  fig3-34.c  multi-fork    my_shell.c      newproc-win32.c  shm-posix-consumer.c  simple-shell.c  win32-pipe-child.c
a.out  DateServer.java  fig3-31.c  fig3-33.c  fig3-35.c  multi-fork.c  newproc-posix.c  pid.c           shm-posix-producer.c  unix_pipe.c     win32-pipe-parent.c
exit
```

Remaining problem:

- 使用 & 后台运行一个命令，但之后却一直会不到原来 wait 的状态，后来的命令都同时运行，无 wait?

```
osh>ls
1.txt  DateClient.java  fig3-30.c  fig3-32.c  fig3-34.c  multi-fork    my_shell.c
a.out  DateServer.java  fig3-31.c  fig3-33.c  fig3-35.c  multi-fork.c  newproc-posix.c
osh>ls &
osh>1.txt  DateClient.java      fig3-30.c  fig3-32.c  fig3-34.c  multi-fork    my_shel
a.out  DateServer.java  fig3-31.c  fig3-33.c  fig3-35.c  multi-fork.c  newproc-posix.c
pwd
osh>/home/liukanglai/Learing/Code/CLionProjects/democ/OS/final-src-osc10e/ch3
exit
```

- 如图：第二个 pwd 命令也在提示符后面输出，debug 时未检查出，
- 还是 fork-wait 的问题，内部机制仍不是很清楚