openEuler编译openEuler

可参考的教程:

https://www.cnblogs.com/salty-pineapple/articles/18226917
https://forum.openeuler.org/t/topic/615

在欧拉操作系统上:

编译前置:

1. 拉取对应代码:
   两种方式:

   ```
   -- 这里注意分支名字
   git clone -b OLK-6.6 https://gitee.com/openeuler/kernel.git
   -- 或者这样:
   dnf install -y kernel-source
   ```

2. 安装编译需要的工具:

   ```
   dnf install -y rpm-build openssl-devel bc rsync gcc gcc-c++ flex bison m4
   elfutils-libelf-devel ncurses-devel
   ```

3. 编辑kernel/Makefile文件:

   ```
   # SPDX-License-Identifier: GPL-2.0
   VERSION = 6
   PATCHLEVEL = 6
   SUBLEVEL = 0
   EXTRAVERSION =
   NAME = Hurr durr I'ma ninja sloth
   ```

   注意这里的版本设置要比当前操作系统的版本高, 所以这里将 EXTRAVERSION 设置为 33
   另外在修改的时候: 需要注意: 在数字后面, 千万不要加空格, 否则会导致编译失败.

补丁

拷贝patch文件

将RePab的三份文件拷贝到kernel/patch(自己创建一个patch文件夹)

打补丁

`patch -p1 <patch/RePABp.patch`

注意这里会有几个报错, 按照rej文件进行修改
在kernel文件夹下修改如下:
mm/memory.c:

```
        }

        /* No need to invalidate - it was non-present before */
        update_mmu_cache_range(vmf, vma, address, ptep, nr_pages);
#ifdef CONFIG_REPABP
        repabp_register(vmf->address,&folio->page);
#endif
unlock:
        if (vmf->pte)
                pte_unmap_unlock(vmf->pte, vmf->ptl);
out:
        /* Clear the swap cache pin for direct swapin after PTL unlock */
        if (need_clear_cache)
                swapcache_clear(si, entry, 1);
        if (si)
                put_swap_device(si);
        return ret;
```

mm/Makefile:

```
obj-$(CONFIG_SHRINKER_DEBUG) += shrinker_debug.o
obj-$(CONFIG_SHARE_POOL) += share_pool.o
obj-$(CONFIG_REPABP) += repabp/
obj-$(CONFIG_MEMCG_MEMFS_INFO) += memcg_memfs_info.o
obj-$(CONFIG_ETMEM) += etmem.o
obj-$(CONFIG_PAGE_CACHE_LIMIT) += page_cache_limit.o
obj-$(CONFIG_CLEAR_FREELIST_PAGE) += clear_freelist_page.o
obj-$(CONFIG_MEMORY_RELIABLE) += mem_reliable.o
```

编译:

```
cd kernel/ --进入前面下载下来的kernel目录
make openeuler_defconfig -- 生成默认的.config文件
make menuconfig -- 进行配置, 选择需要编译的模块
make binrpm-pkg -j8
```

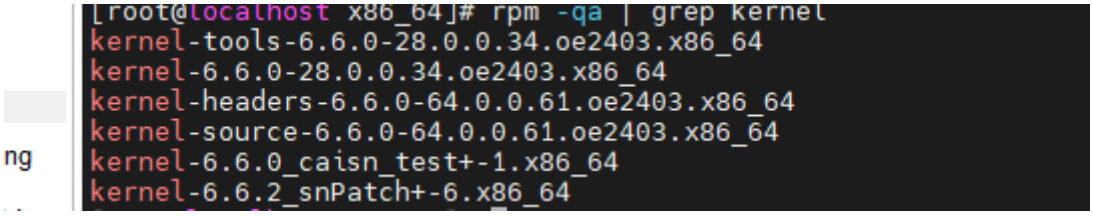
注意: 在make menuconfig的时候, 可以设置下下面这里:
General setup -> Local version - 随便填入一些字, 但需要以"-"开头, 最终生成的内核名字可以在kernel/rpmbuild/RPMS/x86_64/下看到

```
.config - Linux/x86 6.6.0 Kernel Configuration
> General setup
                                General setup
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for
  Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

      [ ] Compile also drivers which will not load
      [ ] Compile the kernel with warnings as errors
      (-sinantest) Local version - append to kernel release
      [ ] Automatically append version information to the version string
      ()  Build ID Salt
          Kernel compression mode (Gzip)  --->
      ()  Default init path
      ((none)) Default hostname
      [*] System V IPC
      [*] POSIX Message Queues
      [ ] General notification queue
      [*] Enable process_vm_readv/writev syscalls
      [ ] uselib syscall (for libc5 and earlier)
      -*- Auditing support
          IRQ subsystem  --->
```

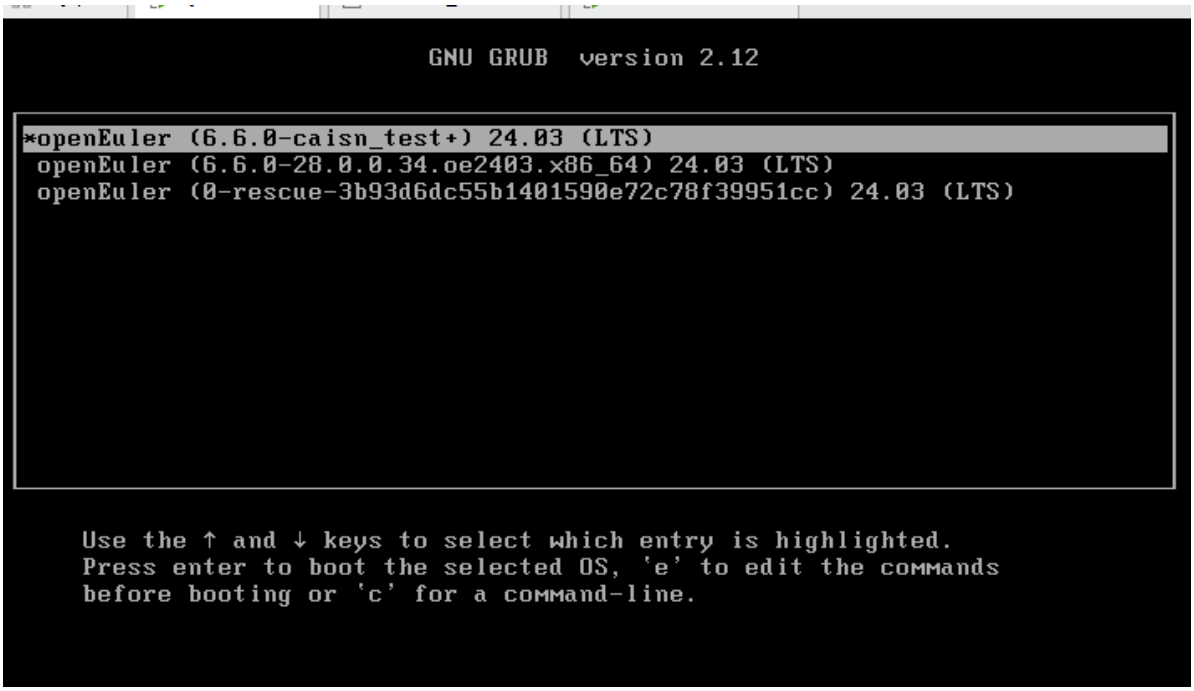
安装:

```
cd kernel/rpmbuild/RPMS/x86_64/
rpm -ivh kernel-6.6.0-1.oe1.x86_64.rpm -- 安装内核
![alt text](image-1.png)
查看安装的内核:
rpm -qa | grep kernel
```



加载:

重启机器, 在进入内核选择页面的时候, 选择编译安装的内核



通过uname -r进行查看当前系统的内核版本