

从Unikernel到宏内核

M.1.0 UserPrivilege

- 宏内核特点
- 组件化的使用——隔离的策略
- 课后练习——缺页异常

M.1.0 UserPrivilege

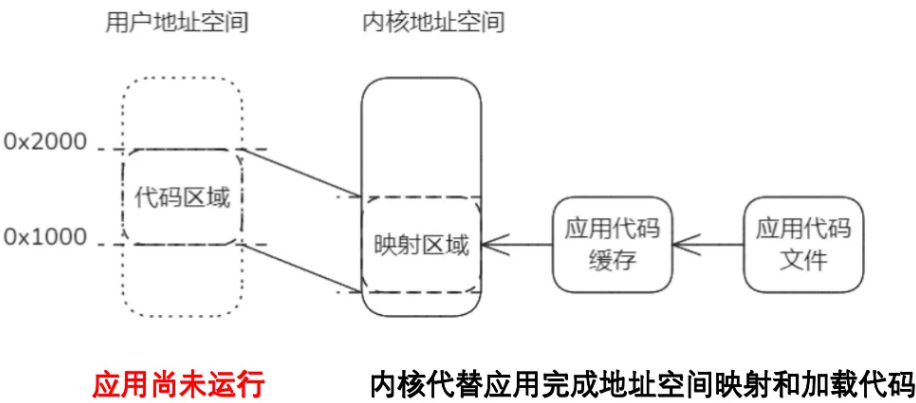
- 从Unikernel扩展为宏内核的基本条件：特权级、地址空间、系统调用等
- 用户特权级和特权级上下文切换机制

宏内核特点

- 增加一个权限较低的用户特权级来运行应用。
- 为应用创建独立的用户地址空间，与内核隔离
- 内核运行时可以随时加载应用Image投入运行

把应用加载到用户地址空间

- 第一步，从文件加载代码到内存缓冲区。
- 第二步，为用户地址空间代码区域建立映射，拷贝代码到被映射页面中。




- 应用与内核界限分明。

组件化的使用——隔离的策略

- 调度器只关注任务的调度属性
- 资源属性由ext拓展来实现
- 这样可以实现调度器多种内核类型的复用

课后练习——缺页异常


- 较为简单，主要就是去关注一些代码的写法，比如 `linkme::distributed_slice`来存储分布的静态切片来构建中断向量表，，以及学习一些宏的写法，唯一遇到的问题是没做make clean，导致一开始修改为false的时候还能跑通，make clean之后就对了

tour > m_1_0 > src >  syscall.rs >  handle_page_fault

```
13 fn handle_syscall(tf: &TrapFrame, syscall_num: usize) -> isize {
14     let ret: isize = match syscall_num {
23         ...
24     };
25     ret
26 }
27
28 #[register_trap_handler(PAGE_FAULT)]
29 fn handle_page_fault(vaddr: VirtAddr, flags: MappingFlags, is_usr: bool) -> bool {
30     ax_println!("handle_pagefault ...");
31     if is_usr {
32         if !axtask::current().CurrentTask
33             .task_ext() & TaskExt
34             .aspace Arc<Mutex<AddrSpace>>
35             .lock() MutexGuard<'_, AddrSpace>
36             .handle_page_fault(vaddr, access_flags: flags)
37         {
38             ax_println!("{}", segmentation fault, exit!", axtask::current().id_name);
39             axtask::exit(exit_code: -1);
40         } else {
41             ax_println!("{}", handle page fault OK!", axtask::current().id_name);
42         }
43         true
44     } else {
45         false
46     }
47 }
48
```

问题 3 输出 终端 GITLENS

> > v 终端

+ v  bash  

```
arch = riscv64
platform = riscv64-qemu-virt
target = riscv64gc-unknown-none-elf
smp = 1
build_mode = release
log_level = warn

[ 1.153152 0 fatfs::dir:139] Is a directory
[ 1.239093 0 fatfs::dir:139] Is a directory
[ 1.294890 0 fatfs::dir:139] Is a directory
[ 1.352661 0 fatfs::dir:139] Is a directory
app: /sbin/origin
paddr: PA:0x80643000
Mapping user stack: VA:0x3fffff0000 -> VA:0x4000000000
New user address space: AddrSpace {
  va_range: VA:0x0..VA:0x4000000000,
  page_table_root: PA:0x80642000,
}
Enter user space: entry=0x1000, ustack=0x4000000000, kstack=VA:0xfffffc080688010
handle_pagefault ...
Task(4, "userboot"): handle page fault OK!
handle_syscall ...
[SYS_EXIT]: process is exiting ..
monolithic kernel exit [Some(0)] normally!
zjz@zjz:~/arceos/oscamp/arceos$
```

