Lab4 实验报告

周美廷 | 76066002

# **思考4.1**

思考并回答下面的问题：

• 内核在保存现场的时候是如何避免破坏通用寄存器的？

• 系统陷入内核调用后可以直接从当时的 $a0-$a3 参数寄存器中得到用户调用 msyscall 留下的信息吗？

• 我们是怎么做到让 sys 开头的函数“认为”我们提供了和用户调用 msyscall 时 同样的参数的？

• 内核处理系统调用的过程对 Trapframe 做了哪些更改？这种修改对应的用户 态的变化是？

# **思考4.2**

思考下面的问题，并对这两个问题谈谈你的理解：

• 子进程完全按照 fork() 之后父进程的代码执行，说明了什么？

• 但是子进程却没有执行 fork() 之前父进程的代码，又说明了什么？

# **思考4.3**

关于 fork 函数的两个返回值，下面说法正确的是： A、fork 在父进程中被调用两次，产生两个返回值 B、fork 在两个进程中分别被调用一次，产生两个不同的返回值 C、fork 只在父进程中被调用了一次，在两个进程中各产生一个返回值 D、fork 只在子进程中被调用了一次，在两个进程中各产生一个返回值

# **思考4.4**

如果仔细阅读上述这一段话, 你应该可以发现, 我们并不是对所有的 用户空间页都使用 duppage 进行了保护。那么究竟哪些用户空间页可以保护，哪些 不可以呢，请结合 include/mmu.h 里的内存布局图谈谈你的看法。

# **思考4.5**

在遍历地址空间存取页表项时你需要使用到 vpd 和 vpt 这两个“指 针的指针”，请思考并回答这几个问题：

• vpt 和 vpd 的作用是什么？怎样使用它们？

• 从实现的角度谈一下为什么能够通过这种方式来存取进程自身页表？

• 它们是如何体现自映射设计的？

• 进程能够通过这种存取的方式来修改自己的页表项吗？

# **思考4.6**

page\_fault\_handler 函数中，你可能注意到了一个向异常处理栈复制 Trapframe 运行现场的过程，请思考并回答这几个问题：

• 这里实现了一个支持类似于“中断重入”的机制，而在什么时候会出现这种 “中断重入”？

• 内核为什么需要将异常的现场 Trapframe 复制到用户空间？

# **思考4.7**

到这里我们大概知道了这是一个由用户程序处理并由用户程序自身来 恢复运行现场的过程，请思考并回答以下几个问题：

• 用户处理相比于在内核处理写时复制的缺页中断有什么优势？

• 从通用寄存器的用途角度讨论用户空间下进行现场的恢复是如何做到不破坏 通用寄存器的？

# **思考4.8**

请思考并回答以下几个问题：

• 为什么需要将 set\_pgfault\_handler 的调用放置在 syscall\_env\_alloc 之前？

• 如果放置在写时复制保护机制完成之后会有怎样的效果？

• 子进程需不需要对在 entry.S 定义的字 \_\_pgfault\_handler 赋值？