操作系统实验文档

尹哲晖 2100012821

一. 开始阶段：对评测平台的适配

在本阶段的开发过程中，我主要依据了 xv6-k210 的代码和相关 PPT 资料进行实现。首先，更新了 Makefile 以定义“all”目标的编译规则，为了适配 qemu 运行环境对一些指令进行修改，包括删除 sudo，修改 sdimg 的大小以及其余一些参数。将测试样例挂载在 fs 中实现可通过 makelocal 指令完成本地测评。

接下来，我对 init.c 进行了调整，以适配大赛要求，具体来说即是要自动运行 32 个评测程序，并在完成所有评测后退出。将该文件编译后以 16 进制格式放置在用户执行的代码所在位置，即 kernel/proc.c 中的 initcode.h 处。具体来说分为两步，先用 runtest.sh 脚本实现将文件编译转换后放置在 kernel/include/initcode.h 处，之后修改 uchar initcode[] = { #include "./include/initcode.h" }; 即可。

通过在 kernel/proc.c 中包含 initcode.h，我们能够在 userinit()函数内部加载并执行这些十六进制编码的指令，从而使内核启动后直接运行预设的测试案例。至此，第一阶段的任务已经完成，内核已能在指定平台上正常运作。接下来，我将根据系统调用文档和测试样例进一步实现具体功能。

二. 执行阶段：功能实现

过程中涉及到系统调用号的修改，参考 https://github.com/oscomp/testsuits-for-oskernel/blob/main/oscomp_syscalls.md 来修改 sysnum.h 以及 syscall.c 文件。

1. sys_read ，sys_write，sys_close 和 sys_fstat

sys_read: 从一个文件描述符中读取

sys_write: 从一个文件描述符中写入

sys_close: 关闭一个文件描述符

sys_fstat: 获取文件状态

实现皆与与 K210 代码基本保持一致，稍作适配上的修改即可。

2. sys_chdir, sys_pipe, sys_dev, sys_readdir, sys_rename 和 sys_remove

sys_chdir: 切换工作目录

sys_pipe: 创建管道

sys_dev, sys_readdir, sys_rename 和 sys_remove: 辅助函数

实现皆与 K210 代码基本保持一致。

3. sys_fork, sys_exit, sys_wait, sys_kill, sys_exec, sys_shutdown

sys_fork: 创建当前进程的一个副本。

sys_exit: 终止当前进程。

sys_wait: 等待子进程终止。

sys_kill: 向指定进程发送信号以终止它。

sys_exec: 用新的程序替换当前进程的内存映像。

sys_shutdwon: 调用 sbishutdown 关闭

实现皆与 K210 代码基本保持一致。

4. sys_getppid 和 sys_getpid

sys_getppid: 返回当前进程的父进程 ID。

sys_getpid: 返回当前进程的进程 ID。

根据已实现的 proc 结构体中内容，myproc()->pid 和(myproc()->parent)->pid 即可。

5. sys_times, sys_uptime 和 sys_gettimeofday

sys_uptime: 返回系统自启动以来的运行时间。

sys_gettimeofday: 获取当前时间并返回。

sys_times: 返回进程的时间信息。

uptime 在 K210 已经实现。sys_gettimeofday 获取当前时间的计数值，将其转换为秒和纳秒，并复制到用户空间指定地址。sys_times 获取当前时间的计数值，将其转换为用户时间、系统时间等，并复制到用户空间指定地址。

6. sys_mkdir 和 sys_mkdirat

sys_mkdir: 在指定路径创建一个新目录。

sys_mkdirat: 在指定文件描述符和路径下创建一个新目录。

sys_mkdir 已在 K210 实现，而 sys_mkdirat 参考 sys_mkdir 函数实现。区别是

sys_mkdir 函数受限于只能创建相对于当前工作目录的目录并且模式参数固定为0。所以在实现 sys_mkdirat 时需要添加 dirfd 表示要创建的目录所在的目录的文件描述符和 mode 描述文件所有权。

7. sys_sleep 和 sys_nanosleep

sys_sleep: 使当前进程休眠指定的秒数。

sys_nanosleep: 使当前进程休眠指定的秒数和纳秒数。

Sleep 在 K210 已经实现，而 nanosleep 基于 sys_sleep 函数实现，在 kernel/timer.c 中添加纳秒计时。通过读取自开机以来的滴答数判断是否满足指定的睡眠时间。

8. sys_unlinkat

sys_unlinkat：移除指定文件的链接。

类似于 mkdirat，首先获取路径参数并去除末尾的斜杠，然后检查路径是否为当前目录或父目录，如果是则返回错误。接着获取目录项并锁定，检查是否为非空目录，如果是则返回错误。然后锁定父目录并删除目录项，最后解锁并释放目录项，返回成功。

9. sys_getcwd

sys_getcwd 需要在原 K210 的 sys_getcwd 函数基础上进行修改。传入的参数需要增加 size 表示 buf 缓存区的大小，并增加判断缓冲区空间是否足够的功能，最后返回当前工作目录的字符串的指针。

10. sys_wait4 和 sys_waitpid

sys_wait4: 等待指定的子进程终止，并获取其终止状态和资源使用信息。

sys_waitpid: 等待指定的子进程终止，并获取其终止状态。

K210 原先已有类似函数 wait，sys_wait4 的不同在于其只关心指定 pid 的进程而 wait 关心所有子进程。因此 wait4 只需在 wait 的基础上加上一句 pid 是否相等的判定，若不相等则不必等待。Waitpid 基于 wait4 实现

11. sys_brk

sys_brk: 调整当前进程的地址空间大小。

sys_brk 函数获取新地址参数，如果新地址为 0，则返回当前进程的地址空间大小；否则，计算地址空间增量并调整进程地址空间，最后返回成功或失败。。

12. sys_clone

sys_clone: 创建当前进程的一个副本，并将其设置为可运行状态。

实际参考了 fork，包括地址空间、上下文以及文件描述符表的复制，函数首先分配一个新进程结构，然后复制当前进程的页表、陷阱帧、文件描述符表和当前工作目录。接着设置新进程的大小、父进程、追踪掩码和堆栈指针。最后，将新进程设置为可运行状态并返回新进程的 PID。

13. sys_uname

sys_uname 函数将系统信息（如系统名称、节点名称、版本等）复制到用户空间指定地址。

14. sys_yield

sys_yield：让出当前进程的 CPU 使用权，以便调度其他进程运行。

可以通过 K210 原版的 yeild 函数实现。

15. 其他

其他已通过但未提及的均在 K210 已有实现或者通过少量修改即可适配。

三. 总结

在本次实验中，我们主要实现了一个简单的操作系统内核，并建立了各类基础的系统调用功能，使其能够在指定环境中顺利运行。通过详细研究和代码分析 XV6-K210 项目，我们加深了对操作系统核心概念与原理的理解，对操作系统的进程管理、内存管理和文件系统管理等关键模块有了更加具体的认识。在修改和优化 XV6-K210 源代码的过程中，我们也提升了查找问题、解决问题和调试代码的能力。实验中也存在不足，部分调用并未实现或实现错误，例如 test_getdents，test_dup2,test_mmap, test_munmap,test_mount，test_umount 等。