**深 圳 大 学 实 验 报 告**

**课程名称： 操作系统**

**实验项目名称： 综合实验2**

**学院： 计算机与软件学院**

**专业： 计算机科学与技术**

**指导教师： 谭舜泉**

**报告人：熊昕钰、刘玉婷、王志仁 学号：2020150174、2020281053、2020192014 班级： 20级国际班、20级国际班、20级信息与计算科学02班**

**实验时间： 2023年5月9日-2023年6月25日**

**实验报告提交时间： 2023年6月25日**

**教务部制**

|  |
| --- |
| **实验目的与要求：**  **实验目的：**  (1)、掌握计算机操作系统管理进程、处理机、存储器、文件系统的基本方法。  (2)、了解进程的创建、撤消和运行，进程并发执行；自行设计解决哲学家就餐问题的并发线程，了解线程（进程）调度方法；掌握内存空间的分配与回收的基本原理；通过模拟文件管理的工作过程，了解文件操作命令的实质。  (3)、了解现代计算机操作系统的工作原理，具有初步分析、设计操作系统的能力。  (4)、通过在计算机上编程实现操作系统中的各种管理功能，在系统程序设计能力方面得到提升。  **实验要求：**  (1)、题目1：（30%）  阅读xv6中文文档中的Chapter 4 Traps and system calls；  阅读这个很不错的网上课程翻译中的Lec06：  https://mit-public-courses-cn-translatio.gitbook.io/mit6-s081/  结合xv6-riscv代码，回答以下的代码理解问题：  在user/zombie.c中，sleep(5); （L12）。sleep是一个系统调用。请分析代码，阐述在代码中，这一系统调用如何一步步的转化为一个对核心函数sleep（kernel/proc.c / L536）的调用？  (1)、题目2：（70%）  一个在xv6-riscv上的游戏实例（xv6-riscv-2048.zip，a, d, w, s 分别代表左，右，上，下。）：  这个小游戏移植于https://github.com/nyuichi/xv6。  可以读一下这个中文报道：  https://finance.sina.com.cn/tech/2020-10-08/doc-iivhuipp8489972.shtml  但我们只是完成了部分移植。原有版本上还有一些重要的重要功能我们并没有移植到risc-v上。比如原本包含的ioctl系统调用等。Nyuichi的xv6版本上还包含更多有趣的小程序，比如一个迷你的vi，一个扫雷游戏等。  请参考2048的的示例代码，实现一个xv6-riscv上的小游戏，或者小编辑器。  你可以：   * 自己写一个小游戏，或者类似nano的小编辑器；   移植一个开源的小应用程序到xv6-riscv上，比如移植Nyuichi的xv6版本上的mini-vi或扫雷程序到xv6-riscv上。  在提交大作业报告的同时，提供一个ZIP文件包，里面**包含完整的源代码以及编译好的XV6虚拟机可执行文件**。  你们对于这个题目不感兴趣，想在大作业中对于xv6-riscv作其他方面的改进也可以。但是要先征得谭舜泉老师的同意。  **说明：**  （1）本次实验课作业满分为100分，占总成绩的比例（待定）。  （2）本次实验课作业截至时间2023年6月25日（周日）23:59。  （3）个人信息：WORD文件名中的“姓名”、“学号”，请改为你的姓名和学号；实验报告的首页，请准确填写“学院”、“专业”、“报告人”、“学号”、“班级”、“实验报告提交时间”等信息。  （4）实验报告中的图表规范要注意，特别是图号、图题和表号、表题不要缺漏。否则扣除相应的格式分。  （5）提交方式：请在BLACKBOARD平台中按时提交；延迟提交不得分。  （6）发现抄袭（包括复制&粘贴整句话、整张图），该次作业记零分。  （7）期末考试阶段补交无效。 |

|  |
| --- |
| **(1)、题目1：**回答以下的代码理解问题：  在user/zombie.c中，sleep(5); （L12）。sleep是一个系统调用。请分析代码，阐述在代码中，这一系统调用如何一步步的转化为一个对核心函数sleep（kernel/proc.c / L536）的调用？  答：  1. 用户程序通过中断（例如，软中断）进入内核空间，中断处理程序根据系统调用号将控制权传递给相应的系统调用处理函数。  2. 在xv6系统中，系统调用处理函数位于sysproc.c文件中。当系统调用号为SYS\_sleep时，控制权将传递给sys\_sleep函数。    图1.sys\_sleep函数  3. sys\_sleep函数将用户程序传递的参数（这里是5）从用户空间拷贝到内核空间，并调用内核中的核心函数sleep。    图2.调用核心函数sleep  4. 核心函数sleep位于proc.c文件中。它负责在当前进程的状态中设置睡眠时间，并将该进程转移到睡眠队列中等待一段时间。    图3.sleep函数  5. sleep函数在设置睡眠时间后，调用另一个内核函数sched，该函数负责选择下一个要运行的进程。    图4.sched函数  6. 在sched函数中，调度算法会选择一个新的可运行进程（可能是睡眠时间到了的进程），并将控制权切换到该进程，使其继续执行。    图5.选择一个新的可运行进程  综上所述，sleep(5);系统调用在XV6系统中被转化为一个对sleep核心函数的调用。该核心函数负责设置当前进程的睡眠时间，并通过调度算法选择下一个要运行的进程。  **(2)、题目2：**请参考2048的的示例代码，实现一个xv6-riscv上的小游戏，或者小编辑器。  答：  我们移植的是扫雷游戏，以下是我们实现的过程，分为代码实现和移植程序两个部分。 代码实现  1. **修改头文件**   移植前的代码使用的是C标准库头文件，我们将其更改为xv6-riscv-2048中自带的头文件实现移植。    图1.修改头文件   1. **设置游戏界面参数**   定义扫雷游戏的界面长宽和cell的标记信息。    图2.设置游戏界面参数   1. **展示界面**   面板的数据结构是二维数组，其中数组的每个元素是一个有特殊含义的值，和上一节定义的符号一致，因此和宏定义的值进行与运算即可判断出值的含义，如果为1就是某个宏定义的含义，否则不是。   * 1. board2disp函数返回的值是0~3，代表简化后的值。     图3-1.board2disp函数   * 1. showboard函数的作用是打印面板。     图3-2.showboard函数   1. **判断合法性**   通过isvalid函数来判断输入坐标的合法性，进行越界检查。    图4.判断合法性   1. **初始化**    1. 根据难度设置面板大小和地雷数量  * 简单模式的面板大小是9x9，10个地雷。 * 中等模式的面板大小是16x16，40个地雷。     图5-1.初始化面板大小和地雷数量   * 1. 初始化每个cell的地雷数量   将面板每个位置都初始化为0，确保在放置地雷之前，每个单元格的地雷数量都是初始状态。    图5-2.初始化每个cell的地雷数量   * 1. 根据给定的坐标放置地雷   我们使用int2函数根据被顶的坐标(p,q)在游戏板上放置地雷。  当玩家首次点击游戏板时，游戏应该保证玩家点击的位置是安全无雷的。因此在初始化游戏板并防止地雷时，需要避免在(p,q)及其周围的位置放置地雷。具体的放置过程如下：   * 检查面板的尺寸是否足够容纳放置的地雷数量，不够则返回。 * 循环放置，直到放置的地雷数量达到目标值。 * 随机生成一个(x,y)，检查合法性后，检查位置(x, y)是否位于(p-1, q-1)到(p+1, q+1)的范围内，如果是，则继续下一次循环，避免在初始位置附近放置地雷。 * 检查位置 (x, y) 是否已经放置了地雷，如果是，则继续下一次循环，避免在同一位置重复放置地雷。 * 将位置 (x, y) 的值设置为地雷 (BOMB)。 * 使用嵌套循环，更新周围 8 个单元格的周边地雷数量，将地雷数量存储在低 4 位。     图5-3.根据给定的坐标放置地雷   1. **检查是否获胜/游戏结束**   我们使用isclear函数来检查游戏板上的方块是否全部被揭开或标记为地雷。如果循环遍历结束后没有找到未被揭开且不是地雷的方块，即所有方块都被正确揭开或标记为地雷，函数会返回 1，表示游戏胜利。    图6-1.isclear函数  反之，如果拆出来的是地雷，使用isgameover函数来表示玩家失败了，游戏结束。    图6-2.isgameover函数   1. **打开cell**   我们使用mine\_open函数打开cell。  函数首先声明了一些变量，包括整型变量 x 和 y，用于存储要打开的方块的坐标，以及一个临时的 command 结构体 com\_tmp 和一个整型数组 drc，用于表示相邻方块的偏移量。  然后，函数检查指定坐标处的方块是否已经被打开或标记。如果方块未被打开或标记（即未设置 OPENED 或 MARKED 标志位），则将 OPENED 标志位置为1，表示该方块已被打开。否则，函数直接返回，不做任何操作。  接下来，函数检查指定坐标处的方块是否为零，即不包含地雷。如果是零，则遍历方块周围的8个相邻方块。对于每个相邻方块，创建一个临时的 command 结构体 com\_tmp，将其坐标设置为相应的偏移量。  最后，递归调用 mine\_open 函数，以打开相邻方块：当点击的方块是空白方块时，会自动打开周围的空白方块。    图7.打开cell   1. **标记cell**   我们使用mine\_mark函数检查指定坐标处的方块是否已经被打开。如果方块未被打开（即未设置 OPEN 标志位），则使用按位异或运算符 ^ 将 MARKED 标志位进行取反操作，以实现标记和取消标记的切换。    图8.标记cell   1. **实现扫雷逻辑的主函数**    1. 首先，如果用户输入了初始化的指令，就初始化面板。     图9-1.根据用户指令初始化面板   * 1. 然后调用action函数，判断用户的操作，并调用相应的函数，比如是打开cell还是标记cell，以及退出、初始化等。     图9-2.action函数   * 1. 在用户每一步操作后都进行判断，是否退出游戏，游戏是否获胜/失败。     图9-3.对用户操作进行判断 移植程序 实现了代码之后，我们要将小游戏正确移植到xv6系统中。   1. **在makefile中声明minesweeper文件**     图9.声明minesweeper文件   1. **在makefile中添加如下代码实现编译链接**     图11.在makefile中添加代码  具体作用如下：   1. 编译链接：通过该代码，$U/minesweeper.o 对象文件与 $L/libc.o 和 $U/usys.o 这些依赖项一起被链接，生成了可执行文件 $U/\_minesweeper。 2. 设置入口点：使用 -e main 选项，指定程序的入口点为 main 函数。这是因为操作系统需要知道程序的起始点以正确运行。 3. 指定代码段位置：通过 -Ttext 0 选项，将程序的代码段放置在地址 0 处。在 xv6 操作系统中，程序的代码段通常从地址 0 开始。 4. 反汇编生成汇编代码：最后一行使用 $OBJDUMP 工具和 -S 选项，将生成的可执行文件 $U/\_minesweeper 进行反汇编，生成相应的汇编代码，并将其保存在 $U/minesweeper.asm 文件中。这对于分析和调试程序非常有用，可以帮助理解程序的内部执行逻辑。 5. **设置权限**   当我们声明并且编译链接了程序后想要运行，却发现出现了Permission denied的报错，这是由于当前目录的权限为不可执行。  于是我们使用sudo chmod -R 777 mkfs命令，递归地将目录 mkfs 及其所有子目录和文件的权限设置为最高权限，使所有用户都具有读、写和执行的权限。    图12.设置权限为最高权限  值得注意的是，赋予这样高级别的权限可能会导致安全风险，因为我们只是需要执行一下我们移植的程序，而且整个系统也只有cynthia一个用户，所以才选择了这样做。在实际使用中应该谨慎操作，并仅在有必要的情况下进行权限更改。 结果演示  1. 开始      1. Help命令      1. Init难度选择      1. Open扫雷      1. Mark标记   X代表标记。     1. 扫雷失败   B代表bomb。     1. Q退出     ++++++++++++++++++++++++++++++++++++++++++++++++++++++  **其他（例如感想、建议等等）。**  通过参与该实验，我深入了解了计算机操作系统的重要组成部分，包括进程管理、处理机调度、存储器管理和文件系统操作。我学会了创建、撤销和运行进程，并了解到进程可以并发执行的机制。此外，我还通过自行设计解决哲学家就餐问题的并发线程，对线程调度方法有了更深入的了解。在内存管理方面，我掌握了分配和回收内存空间的基本原理，这对于优化程序性能和资源利用至关重要。  在模拟文件管理的工作过程中，我了解了文件操作命令的实质，从而更好地理解了文件系统的工作原理和重要性。通过这些实践，我对现代计算机操作系统的工作原理有了更清晰的认识，并且培养了初步分析和设计操作系统的能力。  最重要的是，通过在计算机上编程实现操作系统中各种管理功能，我提高了自己的系统程序设计能力。这种实践让我更加熟悉操作系统的内部机制，并能够以更高效和可靠的方式编写系统级的程序。  总之，这次实验使我对计算机操作系统有了更全面的了解，提升了我的技能和能力。我相信这些所学将对我今后的学习和职业发展产生积极的影响。 |

深圳大学学生实验报告用纸

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：谭舜泉  2023年 7 月 2 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。