**操作系统**

**实 验 报 告**

实验名称： 综合实验2: 系统调用扩充

学 员： 左骁 学 号： 202002001070

培养类型： 无军籍 年 级： 2020级

专 业： 计算机类 所属学院： 计算机学院

指导教员： 文艳军 职 称： 教授

实 验 室： 306-704 实验日期： 2022-06-26

国防科学技术大学训练部制

《实验报告》填写说明

1．学员完成人才培养方案和课程标准要所要求的每个实验后，均须提交实验报告。

2．实验报告封面必须打印，报告内容可以手写或打印。

3．实验报告内容编排及打印应符合以下要求：

（1）采用A4（21cm×29.7cm）白色复印纸，单面黑字打印。上下左右各侧的页边距均为3cm；缺省文档网格：字号为小4号，中文为宋体，英文和阿拉伯数字为Times New Roman，每页30行，每行36字；页脚距边界为2.5cm，页码置于页脚、居中，采用小5号阿拉伯数字从1开始连续编排，封面不编页码。

（2）报告正文最多可设四级标题，字体均为黑体，第一级标题字号为4号，其余各级标题为小4号；标题序号第一级用“一、”、“二、”……，第二级用“（一）”、“（二）” ……，第三级用“1.”、“2.” ……，第四级用“（1）”、“（2）” ……，分别按序连续编排。

（3）正文插图、表格中的文字字号均为5号。

一、实验目的和内容

**实验目的：**

深入掌握操作系统内核程序开发方法。 实验内容：以版本 0 内核为基础，增加一组系统调用（详情如下），并通过给定的测试用例。

图形用户界面, 应用程序, Word

描述已自动生成

要求提交实验报告和所有源码，实验报告应记录各系统调用的设计思路、实现方法和测试过程及画面。

1. 操作方法与实验步骤

**1、添加系统调用号**

在内核代码中添加系统调用号和函数的声明。

1）在unistd.h中添加系统调用号。

文本

描述已自动生成

1. 在sys.h中添加函数定义，修改sys\_call\_table，在对应系统调用号处添加新定义的系统调用函数。

文本

描述已自动生成



3）在system\_call.s文件中修改nr\_system\_calls数量的宏定义。



**2、添加实现函数**

①getcwd

函数功能：获取当前⼯作⽬录

实现思路：getcwd()函数将内含当前工作目录决定路径的字符串(包括结尾空字符)置于buf指向的已分配缓冲区中。调用者必须为buf缓冲区分配至少size个字节的空间（通常，buf的大小与PATH\_MAX常量相当）一旦调用成功，getcwd()将返回一枚指向buf的指针。如果当前工作目录的路径名长度超过size个字节，那么getcwd()会返回 NULL，并将 errno 置为 ERANGE。若 cwdbuf 为 NULL，且 size 为 0，则 glibc 封装函数会为 getcwd()按需分配一个缓冲区，并将指向该缓冲区的指针作为函数的返回值。为避免内存泄漏，调用者之后必须调用 free()来释放这一缓冲区。如果 buf 是 NULL，那么 getcwd()将分配一个大小为 size 字节的缓冲区，用于向调用者返回结果。glibc 的 getcwd()也实现了这一特性。

文本

描述已自动生成

文本

描述已自动生成

②sleep

函数功能：进程挂起一段确定时间后继续重新调度并且运行.

实现思路：实现这个函数主要分为以下三步：a、注册一个信号signal (SIGALRM,handler)。接收内核给出的一个信号。

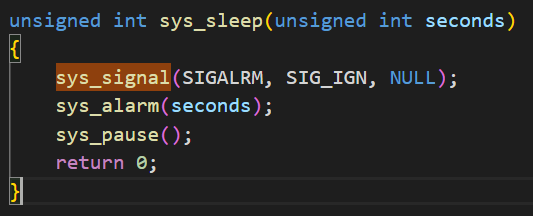
b、调用alarm()函数。

alarm()函数的主要功能是设置信号传送闹钟，即用来设置信号SIGALRM在经过参数seconds秒数后发送给目前的进程。如果未设置信号SIGALARM的处理函数，那么alarm()默认处理终止进程。函数返回值：如果在seconds秒内再次调用了alarm函数设置了新的闹钟，则后面定时器的设置将覆盖前面的设置，即之前设置的秒数被新的闹钟时间取代；当参数seconds为0时，之前设置的定时器闹钟将被取消，并将剩下的时间返回。

c、pause()挂起进程。

当执行到这个函数的时候，当前进程被挂起，等时钟alarm函数到时间之后，内核发送一个SIGALRM信号。导致控制从pause函数转到信号的处理函数。信号处理函数中的代码被执行，然后控制返回。当信号被处理完毕之后，pause函数返回，进程继续。

了解了各个函数函数的功能就可以按照sleep的定义写出代码

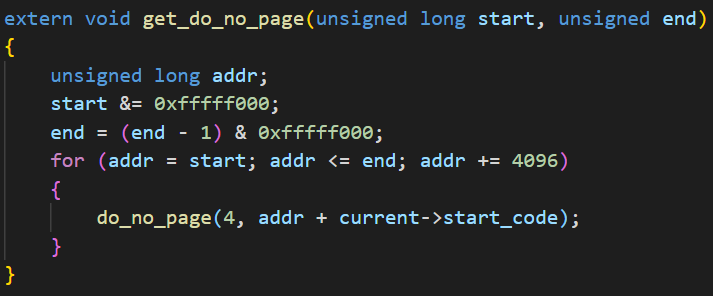


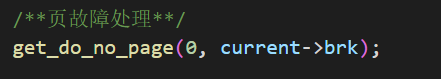
对于sys\_signal函数的参数解释：如果仅仅使用sys\_alarm唤醒,该函数会向系统发出一个软中断信号SIGALRM,来使得操作系统中断掉这个进程,因此我们需要屏蔽掉这个信号产生的效果.

③execve2

函数功能：以立即加载方式执行一个指定的程序。此系统调用结束后，进程运行时不应再发生代码段和数据段中的缺页故障。

实现思路：该函数主要是修改do\_execve使其可以进行页故障处理，这可以通过do\_no\_page函数实现。编写一个get\_do\_no\_page函数调用do\_no\_page，再在do\_execve2中调该函数强制进行也故障处理。通过分析do\_execve代码，发现current->brk记录的是要加载的大小，因此get\_do\_no\_page的参数可以传入0和current->brk

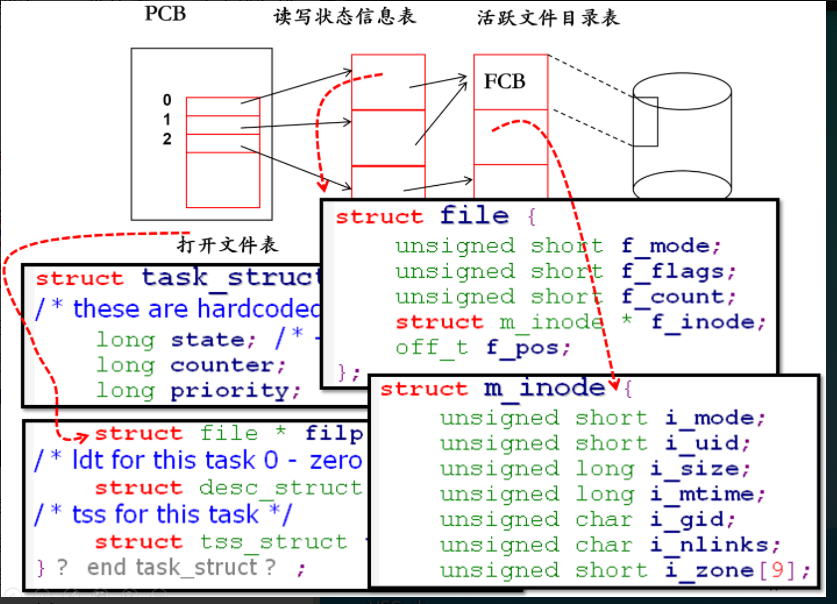




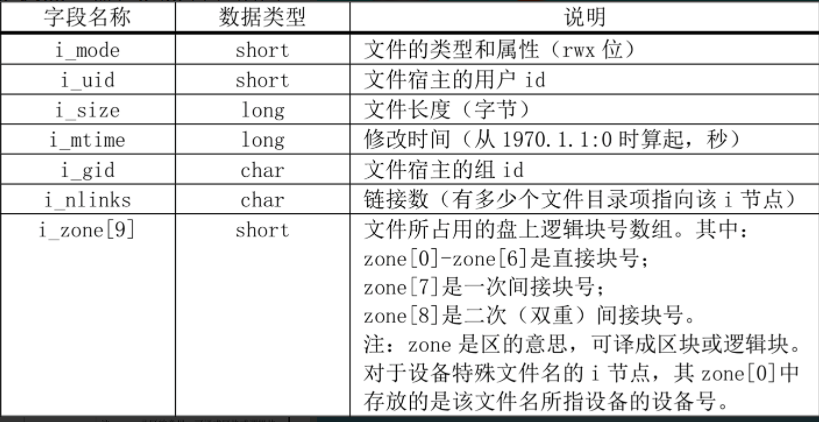
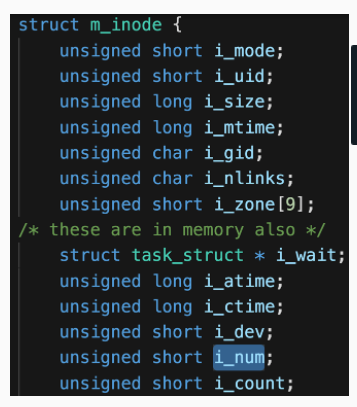
④int getdents(unsigned int fd, struct linux\_dirent \*dirp, unsigned int count);

函数功能：获取目录的目录项

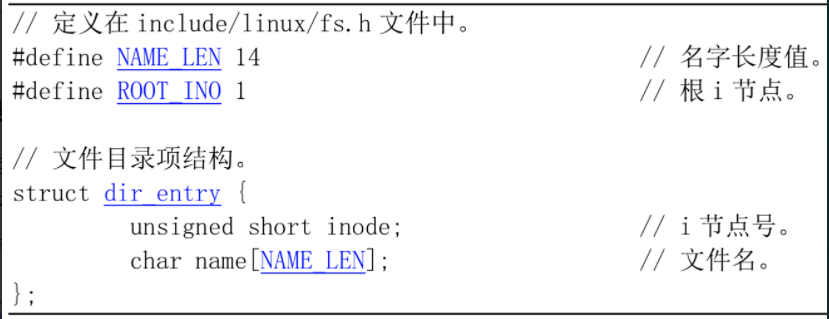
实现思路：首先明确函数每个参数的含义。fd是所要读取目录的文件描述符，故可以访问filp数组得到获得操作文件。在这里需要注意越界问题，即判断fd是否小于NR\_OPEN。dirp是定义的一个struct linux\_dirent结构型指针，是用于保存所读取目录信息的缓冲区。count是缓冲区的大小，在代码的实现中常常用于判断缓冲区溢出问题。在函数中我们还调用了fs.h库中定义的一系列结构体，在这里用课堂上老师讲授相关的内容作出解释。



NOTICE：f\_inode是包含扩展信息索引节点。下面给出m\_inode结构体的定义以及解释。

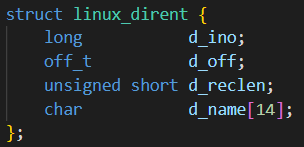


以下是目标文件信息，用于读取目录项。

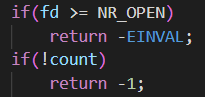


函数实现过程

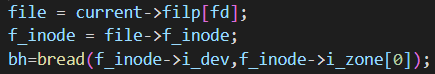
1. 定义struct linux\_dirent



1. 定义一系列局部变量以及判断是否越界。

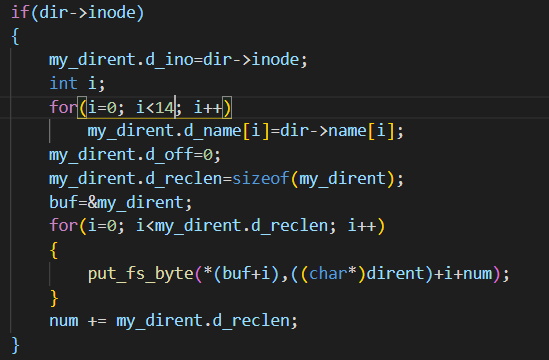


1. 获取有效信息。



首先用fd访问filp数组得到操作文件file；其次访问file得到索引节点f\_inode；然后通过调用bread块设备读取获取该文件数据对应硬盘区域的内容，存入struct buffer\_head结构性指针bh中。

1. 遍历文件下的目录项，存入缓冲区drip。



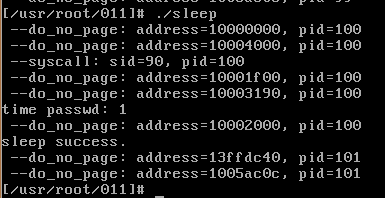
根据格式要求，对结构体成员进行赋值。最后需要使用put\_fs\_byte函数将内核态数据传入用户态。由于函数是返回读取字节数，因此我们需要在遍历过程中对定义的int类型变量num进行更新，用num来记录目录项长度。

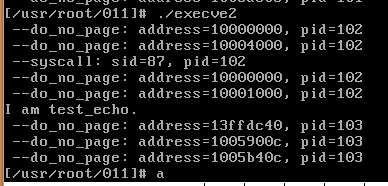
1. 利用检验代码，在虚拟机下检验是否改写成功
2. 实验结果与分析

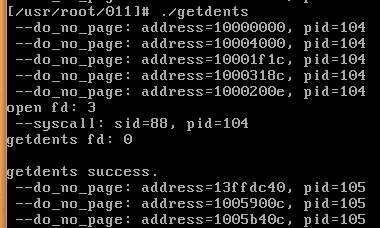
利用检验代码，每个函数在虚拟机下均通过检验。运行截图如下：

文本

描述已自动生成







四、问题与建议(可选)

问题：在虚拟机中运行时出现bash: ./build.sh: Permission denied的报错

原因：文件为只读，或者用户没有权限。

修改方法：运行命令： chmod 777 build.sh

问题：在虚拟机中运行时出现—do\_no\_page的地址与调用号

原因：补丁删除打印信息的命令

修改方法：卸载补丁 patch -p2 -R <execve2.patch>