操作系统实验一：实验及源码阅读要求

1. 实验考核要求

1.1 实验内容

在 Nachos 内核添加下面五个系统调用：

- int Create(char *name);
- int Open(char *name);
- int Write(char *buffer, int size, int id);
- int Read(char *buffer, int size, int id);
- int Close(int id);

系统调用声明见 code/userprog/syscall.h

1.2 实现要求

1.2.1 Create 系统调用

int Create(char* filename);

描述：当 Create 系统调用触发时，在当前 test 目录下创建一个名为 filename 的文件

返回值：成功时返回 1，失败时返回-1

1.2.2 Open 系统调用

int Open(char* filename);

描述：当 Open 系统调用触发时，在进程中打开名为 filename 的文件，并返回该文件在进程中的 file id，文件只有在被打开后才能读写

返回值：成功时返回文件号 file id，失败时返回-1

1.2.3 Write 系统调用

int Write(char* buffer, int size, int id);

描述：当 Write 系统调用触发时，从内存 buffer 中向编号为 id 的文件中写入 size 个字符

返回值：成功时返回写入的字符数，失败时返回-1

1.2.4 Read 系统调用

int Read(char *buffer, int size, int id);

描述：当 Read 系统调用触发时，从编号为 id 的文件中读取 size 个字符到内存 buffer 中

返回值：成功时返回读出的字符数，失败时返回-1

1.2.5 Close 系统调用

int Close(int fileid);

描述：当 Close 系统调用触发时，关闭编号为 fileid 的文件，文件关闭后不能再读写

返回值：成功时返回 1，失败时返回-1

1.3 实验测试

实验测试时使用提供的**测试代码文件**对每个系统调用进行测试运行，测试代码见下文，或查看 syscallTest.c 文件。

1.3.1 测试代码

syscallTest.c

```
#include "syscall.h"

/*整数转字符串*/
void itoa(int n, char* line);

/*将数字写入文件*/
void writeNum(int num,int fid);

int main(){
	int result,resultReadF;
	int fid1, fid2, fid3;
	char str[64];
	//测试Create和Open
	result = Create("test.txt");//成功创建test.txt文件，返回值为1
	Create("result.txt");//result.txt用来存储返回值
	fid1 = Open("test.txt");//打开存在的文件test.txt，打开成功，返回值为文件id
	fid2 = Open("result.txt");//打开存在的文件result.txt，打开成功，返回值为文件id
	fid3 = Open("fail.txt");  //打开不存在的文件，打开失败，返回值为-1
	//将以上返回值写入result.txt文件
	writeNum(result,fid2);
	writeNum(fid1,fid2);
	writeNum(fid2,fid2);
	writeNum(fid3,fid2);

	//测试Write，向test.txt文件（fid1）写入指定长度数据
	result = Write("SysCall Test for Nachos!\n",25,fid1);//返回值为写入的字符个数
	//将返回值写到result.txt文件
	writeNum(result,fid2);

	//测试close，成功关闭文件，返回值为1
	result = Close(fid1);
	//将返回值写入result.txt文件
	writeNum(result,fid2);

	fid1 = Open("test.txt");
	//测试读Read，从test.txt文件读取指定长度的数据
	result = Read(str,7,fid1); //成功读取数据，返回值为读取的字符个数
	//将返回值写到result.txt文件
	writeNum(result,fid2);
	//将读出的字符串写入result.txt
	Write(str,7,fid2);
	Write("\n",1,fid2);

	//关闭文件
```

```
	Close(fid1);
	Close(fid2);

	Halt();
}

void itoa(int n, char* line){
	int i=0;
	int j=0;
	int neg = -1;
	if(n<0) {
		n=-n;
		neg=1;
	}
	while(n){
		line[i++] = '0'+n%10;
		n/=10;
	}
	if(neg==1) line[i++] = '-';
	for(i=i-1;i>j;j++,i--){
		char t = line[i];
		line[i] = line[j];
		line[j] = t;
	}
}
void writeNum(int num,int fid){
	int cnt = 0;
	int t = num;
	char numStr[64];

	/*计算数字字符串长度*/
	if(t==0) {}
	else {
		if(t<0) cnt++;
		while(t){
			t/=10;
			cnt++;
		}
	}
	itoa(num,numStr);
	Write(numStr,cnt,fid);
	Write("\n",1,fid);
}
```

1.3.2 测试结果

在当前目录 test 下创建了两个文件 test.txt 和 result.txt

test.txt:	result.txt:
Syscall Test for Nachos!	1 6 7 -1 25 1 7 SysCall

示例：

```
wujy@wujy-virtual-machine ~/workspace/nachos_lab/NachOS-4.0/code/test $
wujy@wujy-virtual-machine ~/workspace/nachos_lab/NachOS-4.0/code/test $ ../build.linux/nachos -x syscallTest
open error
open fail.txt failed!
Machine halting!

Ticks: total 1516, idle 0, system 170, user 1346
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
wujy@wujy-virtual-machine ~/workspace/nachos_lab/NachOS-4.0/code/test $ cat test.txt
SysCall Test for Nachos!
wujy@wujy-virtual-machine ~/workspace/nachos_lab/NachOS-4.0/code/test $ cat result.txt
1
6
7
-1
25
1
7
SysCall
wujy@wujy-virtual-machine ~/workspace/nachos_lab/NachOS-4.0/code/test $
```

注意：

- 对文件的具体操作只需调用 nachos 内核实现的接口函数即可，接口函数见 filesys/filesys.h 和 filesys/openfile.h，其中包含两种文件系统的实现，使用 UNIX 实现方式。
- 由于 nachos 内核的文件系统实现非常简单，为了方便实验测试，需要实现对文件的追加写功能，这里修改 filesys/openfile.h 文件中的 Write()函数，如下：

```
int Write(char *from, int numBytes) {
        int currentOffset=Length();//设置当前偏移量为文件中字符长度
        int numWritten = WriteAt(from, numBytes, currentOffset); //从当前偏移量处追加写
        currentOffset += numWritten;
        return numWritten;
        }
```

2. Linux 源码阅读内容及要求

2.1 源码阅读内容

1） Linux 内核整体架构

- 列出 Linux kernel 目录结构
- 说出你认为比较重要的子目录大致负责的功能

2） Linux kernel 的核心模块

- 找到对应的核心源码(具体到哪个目录下的哪个主要的文件)
- 能够简要介绍该模块以及该文件主要做什么

核心模块主要有：

BIOS 启动,内核的加载(这里涉及的汇编不强制要求太多,要知道大概)

进程管理

内存管理

虚拟文件系统

网络子系统

进程间通信

- 稍微深入-数据结构及函数

进程相关

- 进程相关的数据结构-比如 task_struct
- 进程相关的函数-比如进程状态操作相关，fork 相关

2.2 源码阅读报告要求

第一次源码阅读报告主要以了解介绍 Linux 内核的整体架构为主，熟悉内核核心功能模块的主要功能作用、体系结构，源码阅读报告的格式和篇幅与作业模版一致。

3 实验验收与提交

3.1 源码阅读报告和实验报告格式

参考课程主页 http://staff.ustc.edu.cn/~ykli/os/index.html 的作业模版格式

3.2 实验检查时间和地点

时间：最迟 2018 年 4 月 13 日 9:30PM 之前（即当日实验课结束）

地点：电三楼 4 楼教学实验室

3.3 提交要求

实验报告内容主要包含三部分：

1）添加系统调用的主要步骤及核心代码解释说明；

2）实验运行结果截图，并分析说明；

3）实验过程中遇到的问题及解决方法。

实验提交：实验报告+实验代码（包含整个 NachOS-4.0 工程），将实验报告和实验代码一起打包为 zip 压缩包，命名方式“学号+姓名”，提交到 ftp 服务器的“实验提交/实验 1”目录下

源码阅读报告提交：命名方式“学号+姓名”，提交到 ftp 服务器的“源码阅读报告/报告 1”目录下

实验和源码阅读截止日期：2018 年 4 月 13 日 9:30PM 实验课结束