中山大學
SUN YAT-SEN UNIVERSITY

本科生操作系统原理实验报告

题　目：　进程调度算法

学　　号	**21307443**
姓　　名	叶文洁
专业名称	计算机科学与技术
任课教师	刘慈欣
实验地点	实验中心**D501**
实验时间	**2023**年**3**月**1**日—**4**月**1**日

目　录

一、 实验概述

在本次实验中，同学们会熟悉现有Linux内核的编译过程和启动过程，并在自行编译内核的基础上构建简单应用并启动。同时，同学们会利用精简的Busybox工具集构建简单的OS，熟悉现代操作系统的构建过程。此外，同学们会熟悉编译环境、相关工具集，并能够实现内核远程调试。

二、 实验任务

1) 搭建OS内核开发环境包括：代码编辑环境、编译环境、运行环境、调试环境等。
2) 下载并编译i386（32位）内核，并利用qemu启动内核。
3) 熟悉制作initramfs的方法。
4) 编写简单应用程序随内核启动运行。
5) 编译i386版本的Busybox，随内核启动，构建简单的OS。
6) 开启远程调试功能，进行调试跟踪代码运行。

三、 模板使用样例

(1) 插图排版

在“sysucseexp.cls”模板中，除了可以使用普通的figure浮动体环境结合graphicx宏包实现插图排版外，还可以使用floatrow增强浮动体宏包进行浮动体排版(适合多图并排、子图排版等)。有关该宏包的使用细节，请在命令行使用“texdoc floatrow”命令查看其使用说明书。例如，可以用代码1排版图1。

代码清单1: 排版单一插图浮动体

```
\begin{figure}[!htp]
  \begin{floatrow}
    \ffigbox[\FBwidth]{
      \includegraphics[width=0.4\textwidth]{example-image-a}
    }{\caption{一个插图}\label{sharefig:a}}
  \end{floatrow}
\end{figure}
```

当然，也可以用代码2排版带有子图的图2。

代码清单2: 排版单一插图浮动体

```
\begin{figure}[!htp]
  \ffigbox[\textwidth]%
  {%
    \begin{subfloatrow}[2]%\useFCwidth
```

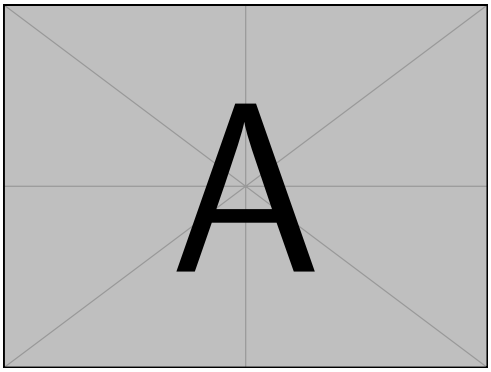


图1 一个插图

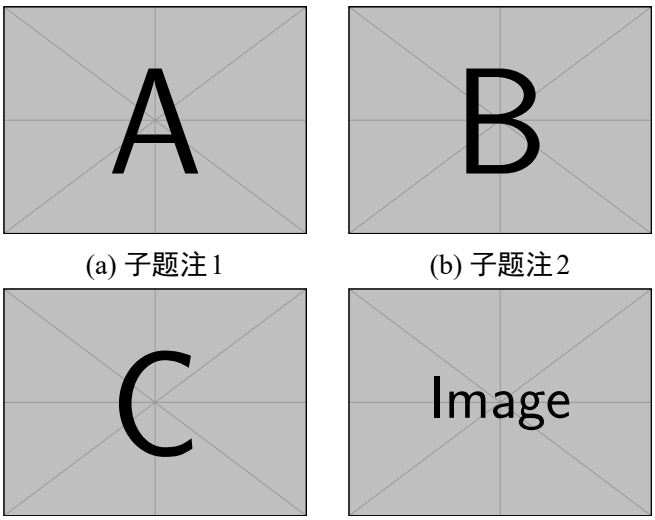
```
      \ffigbox[\FBwidth]{
        \includegraphics[width=0.3\textwidth]{example-image-a}
      }{\caption{子题注1}\label{trifig:a}}
      \ffigbox[\FBwidth]{
        \includegraphics[width=0.3\textwidth]{example-image-b}
      }{\caption{子题注2}\label{trifig:b}}
    \end{subfloatrow}
    \begin{subfloatrow}[2]%\useFCwidth
      \ffigbox[\FBwidth]{
        \includegraphics[width=0.3\textwidth]{example-image-c}
      }{\caption{子题注3}\label{trifig:c}}
      \ffigbox[\FBwidth]{
        \includegraphics[width=0.3\textwidth]{example-image}
      }{\caption{子题注4}\label{trifig:d}}
    \end{subfloatrow}
  }{\caption{四个子图}\label{trifig}}
\end{figure}
```

在模板中，已为插图设置了./figs/、./figure/、./figures/、./image/、./images/、./graphics/、./graphic/、./pictures/、./picture/相对路径，可以在当前工作路径中任意一个这样命名的文件夹，以存放需要的插图文件。如果需要插入一个简单的表格，可以仅使用table和tabular环境实现，如表1。

(2) 表格排版

注意:科技文档中的表格需要采用**三线表**。

如果多个表格布局较为复杂，可以使用floatrow宏包实现排版。如用代码3可编制表2和表3所示的横向并排表格。



(a) 子题注1　(b) 子题注2

(c) 子题注3　(d) 子题注4

图2 四个子图

表1 城市人口数量排名 (source: Wikipedia)

城市	人口
Mexico City	20,116,842
Shanghai	19,210,000
Peking	15,796,450
Istanbul	14,160,467

代码清单3: 表格排版

```
% 横向排版两个表格
\begin{table}[!htp]
  \begin{floatrow}
    \ttabbox[\FBwidth]
    {
      \begin{tabular}{ccl}
        \toprule
        序号 & 测试用例 & \multicolumn{1}{c}{测试目的}\\
        \midrule
        1 & a & 小写字母到大写字母转换\\
        2 & A & 大写字母到小写字母转换\\
        3 & @ & 非字母字符测试\\
        4 & 2 & 非字母其它字符\\
        \bottomrule
      \end{tabular}
    }{\caption{字母大小写转换测试用例表}\label{tab:testsample}}
    \ttabbox[\FBwidth]
    {
```

```
        \begin{tabular}{lr}
          \toprule
          城市 & 人口 \\
          \midrule
          Mexico City & 20,116,842\\
          Shanghai & 19,210,000\\
          Peking & 15,796,450\\
          Istanbul & 14,160,467\\
          \bottomrule
        \end{tabular}
      }{\caption{城市人口数量排名}\label{tab:city2}}
    \end{floatrow}
  \end{table}
```

表2 字母大小写转换测试用例表

序号	测试用例	测试目的
1	a	小写字母到大写字母转换
2	A	大写字母到小写字母转换
3	@	非字母字符测试
4	2	非字母其它字符

表3 城市人口数量排名

城市	人口
Mexico City	20,116,842
Shanghai	19,210,000
Peking	15,796,450
Istanbul	14,160,467

如果表格内容很多，导致无法放在一页内的话，需要用longtable 或longtabu 进行分页。

(3) 项目符号和步骤分点排版

在“sysucseexp.cls”模板中，基于enumitem宏包分别对itemize、enumerate和description三个环境的各个距离参数进行了修正，以使其排版结果符合中文习惯的首行缩进格式。

① itemize环境

- 床前明月光。
- 疑是地上霜。
- 举头望明月。
- 低头思故乡。

② enumerate环境

1) 床前明月光。
2) 疑是地上霜。
3) 举头望明月。
4) 低头思故乡。

③ **description**环境

床前明月光 我欲乘风归去。
疑是地上霜 又恐琼楼玉宇。
举头望明月 高处不胜寒。
低头思故乡 起舞弄清影。

(4) 文本框排版

文本框盒子继承于自己开发的boxie宏包，可用于实验心得总结或注意事项说明。其使用细节请在Github平台查看boxie宏包的使用说明书。同时，在该宏包的基础上，为boxie宏包添加加了摘自于 progartcn 论文模板的“标题”、“注意”、“重要”、“技巧”和“警告”文本框环境代码[1]。

① “标题”文本框

标题文本框环境的使用格式为：

```
\begin{titledBox}{<title>} <content> \end{titledBox}
```

HTTP/Console 内核

HTTP 内核继承自 Illuminate\Foundation\Http\Kernel 类，该类定义了一个 bootstrappers 数组，这个数组中的类在请求被执行前运行，这些 bootstrappers 配置了错误处理、日志、检测应用环境以及其它在请求被处理前需要执行的任务。

② “注意”文本框

注意文本框环境的使用格式为：

```
\begin{noteBox} <content> \end{noteBox}
```

注意

HTTP 内核继承自 Illuminate\Foundation\Http\Kernel 类，该类定义了一个 bootstrappers 数组，这个数组中的类在请求被执行前运行，这些 bootstrappers 配置了错误处理、日志、检测应用环境以及其它在请求被处理前需要执行的任务。

③ “重要”文本框

重要文本框环境的使用格式为：

```
\begin{importantBox} <content> \end{importantBox}
```

重要

HTTP 内核继承自 Illuminate\Foundation\Http\Kernel 类，该类定义了一个 bootstrappers 数组，这个数组中的类在请求被执行前运行，这些 bootstrappers 配置了错误处理、日志、检测应用环境以及其它在请求被处理前需要执行的任务。

[1]本节示例摘自于该模板中的tutorial-sample.tex文件

④ “技巧”文本框

技巧文本框环境的使用格式为：

```
\begin{tipBox} <content> \end{tipBox}
```

技巧

HTTP 内核继承自 Illuminate\Foundation\Http\Kernel 类，该类定义了一个 bootstrappers 数组，这个数组中的类在请求被执行前运行，这些 bootstrappers 配置了错误处理、日志、检测应用环境以及其它在请求被处理前需要执行的任务。

⑤ “警告”文本框

警告文本框环境的使用格式为：

```
\begin{warningBox} <content> \end{warningBox}
```

警告

HTTP 内核继承自 Illuminate\Foundation\Http\Kernel 类，该类定义了一个 bootstrappers 数组，这个数组中的类在请求被执行前运行，这些 bootstrappers 配置了错误处理、日志、检测应用环境以及其它在请求被处理前需要执行的任务。

(5) 补充：模板文件夹中的重要文件

重要

在使用在“sysucseexp.cls”模板前请确保：sysucseexp.cls模板文件在当前工作文件夹中。

如果需要代码盒子等各类盒子排版，则需要确保boxie.sty、fvextra.sty、lstlinebgrd.sty 这3个宏包文件在当前工作文件夹中，并且需要引入boxie宏包，boxie宏包会根据需要加载fvextra和lstlinebgrd 宏包，这两个宏包无需手动加载。

如果需要绘制UML图，则需要加载pgf-umlcd宏包，此时需要确保pgf-umlcd.sty 宏包文件在当前工作文件夹中(TeXLive自带的宏包无法正确处理中文类名称，同时，修改了其association命令，以符合龚晓庆教材绘图习惯)。

如果需要绘制流程图，则需要加载tikz-flowchart宏包，此时需要确保tikz-flowchart.sty 宏包文件在当前工作文件夹中。

如果需要为插图进行标注，则需要加载tikz-imglabels宏包，此时需要确保tikz-imglabels.sty 宏包文件在当前工作文件夹中(TeXLive自带的宏包无法正确标注中的换行问题)。

Makefile文件是执行make命令需要的脚本文件，可以根据需要选择。

.latexmkrc文件是执行latexmk命令需要的脚本文件，可以根据需要选择。

(6) 代码排版

> **重要**
>
> 由于需要排版代码文件，建议使用minted宏包实现代码排版，因此请确保安装有Python及其Pygments模块，详情请使用`texdoc minted`命令查看 minted宏包的使用说明。同时，为支持minted编译，请使用`-shell-escape`编译参数。

可以使用boxie宏包提供的langPyOne、langCVOne等环境或langPyfile和langCVfile等命令实现嵌入式代码或来自文件代码的排版。Py系列环境和命令不带引用计数和标签，CV系列环境和命令带有引用计数和标签。如果需要交叉引用和参考文献，建议按如下方式执行4次编译：

① 使用**minted**宏包排版代码

4 次编译

```
xelatex -shell-escape main.tex
biber main
xelatex -shell-escape main.tex
xelatex -shell-escape main.tex
```

② 使用**listings**宏包排版代码

4 次编译

```
xelatex main.tex
biber main
xelatex main.tex
xelatex main.tex
```

③ 插入行间的命令行代码

命令行代码的插入方式为：`\begin{ubtdark} <content> \end{ubtdark}` 下面提供两个例子：如果需要运行C++测试程序，则执行如下命令：

运行命令

```
make run
```

如果使用`.latexmkrc`，则执行latexmk命令即可：

编译 latex

```
latexmk
```

④ 以嵌入源码的方式插入代码

例如，使用langPyOne环境可以排版不带引用计数和标签的代码。在方框内可指定语言，如果是插入C/C++语言，则参数为C++；如果是插入x86汇编语言，则参数为[x86masm]Assembler。例如下面，插入了一段C++代码：

基于范围的循环

```
int main(){
    // 累加 20 以内的素数
    int sum = 0;
    for(int e : {2, 3, 5, 7, 11, 13, 17, 19}){
        sum += e;
    }
    cout << sum << endl;
    int arr[] = {1, 3, 5, 7, 9};
    // 声明数组 arr,初始化为 5 个奇数
    for(auto &ele : arr){
        // 声明 ele,与数组 arr 关联在一起,用了 auto
        ele = ele * 2;
        // 修改数组每个元素的值
        cout << ele << " ";
        // 输出 ele,2 6 10 14 18
    }
    cout << endl;
    for(auto ele : arr){
        cout << ele << " ";
    }
    // 没有改变:1 3 5 7 9
    cout << endl;
    return 0;
}
```

再如，使用langCVOne环境可以排版带有引用计数和标签的代码，如代码4。

代码清单4：汇编测试程序 asm

```
cin:
  mov ah,0x00  ;从键盘读入字符
  int 16h
  cmp al,0x1b  ;0x1b表示的是esc按键
  je end
  mov ah,0x02  ;移动光标
  int 10h
  mov ah,0x09  ;在当前位置写字符
  int 10h
```

```
  add dl,1
  cmp dl,0x50  ;已经超80列
  je newl
  jmp cin
newl:
  add dh,1     ;换行
  mov dl,0x00
  cmp dh,0x19  ;已经超25行
  je end
  jmp cin
end:
```

⑤ 以源码文件URI的方式插入代码

代码排版命令用于根据代码文件进行排版，因此，可以在IDE中，例如 Code::Blocks 中对代码进行排版，然后将排版后的文件直接用代码排版命令在 LaTeX 中进行排版。例如，使用langPyfile命令可以载入代码文件，排版不带引用计数和标签的代码 (注意指定必要的路径)。

基于范围的循环

```
#include <iostream>
#include <vector>
using namespace std;
int main(){
    // 累加 20 以内的素数
    int sum = 0;
    for(int e : {2, 3, 5, 7, 11, 13, 17, 19}){ // 用 auto 类型更合理
        sum += e;
    }
    cout << sum << endl;
    // 输出结果 77
    int arr[] = {1, 3, 5, 7, 9};
    // 声明数组 arr,初始化为 5 个奇数
    for(auto &ele : arr){
        // 声明 ele,与数组 arr 关联在一起,用了 auto
        ele = ele * 2;
        // 修改数组每个元素的值
        cout << ele << " ";
        // 输出 ele,2 6 10 14 18
    }
    cout << endl;

    for(auto ele : arr){
        cout << ele << " ";
```

```
    }
    // 没有改变:1 3 5 7 9
    cout << endl;
    return 0;
}
```

再如，使用langCVfile命令可以载入代码文件，排版带引用计数和标签的代码 (注意指定必要的路径)，如代码5。

代码清单5: MBR加载程序 asm

```
1  org 0x7c00
2  [bits 16]
3  xor ax, ax
4  ; 初始化段寄存器
5  mov ds, ax
6  mov ss, ax
7  mov es, ax
8  mov fs, ax
9  mov gs, ax
10
11 call read_disk
12 jmp 0x0000:0x7e00
13
14 jmp $
15
16 read_disk:
17         mov ax, 0
18         mov es, ax
19         mov bx, 0x7e00
20         mov ah, 0x02
21         mov al, 0x05
22 times 510-($-$$) db 0
23 db 0x55, 0xaa
```

注：使用minted宏包排版C++代码，可以使用c++/C++或cpp指定语言名称，若使用listings宏包排版代码，则只能使用c++/C++指定语言名称。如果使用TeXstudio、VSCode等IDE工具，请参阅相关资料对IDE进行必要的配置。

四、 实验步骤与实验结果

(1) 实验任务1——复现bootloader加载过程

任务要求

复现”加载bootloader”一节，说说你是怎么做的并提供结果截图，也可以参考Ucore、Xv6等系统源码，实现自己的LBA方式的磁盘访问。

① 步骤一

首先，我们需要安装C++编译工具，安装命令如下：

安装编译工具

```
sudo apt install libssl-dev
sudo apt install libc6-dev-i386
sudo apt install gcc-multilib
sudo apt install g++-multilib
```

② 步骤二

然后，我们需要将内核编译成i386 32位版本。执行命令行如下：

将内核编译成 i386 32 位版本

```
make i386_defconfig
make menuconfig
```

(2) 实验任务2——采用CHS方式读取硬盘

任务要求

在”加载bootloader”一节中，我们使用了LBA28的方式来读取硬盘。此时，我们只要给出逻辑扇区号即可，但需要手动去读取I/O端口。然而，BIOS提供了实模式下读取硬盘的中断，其不需要关心具体的I/O端口，只需要给出逻辑扇区号对应的磁头（Heads）、扇区（Sectors）和柱面（Cylinder）即可，又被称为CHS模式。现在，同学们需要将LBA28读取硬盘的方式换成CHS读取，同时给出逻辑扇区号向CHS的转换公式。最后说说你是怎么做的并提供结果截图。

① 步骤一

首先，我们需要安装C++编译工具，安装命令如下：

安装编译工具

```
sudo apt install libssl-dev
sudo apt install libc6-dev-i386
sudo apt install gcc-multilib
sudo apt install g++-multilib
```

② 步骤二

然后，我们需要将内核编译成i386 32位版本。执行命令行如下：

将内核编译成 i386 32 位版本

```
make i386_defconfig
make menuconfig
```

(3) 实验任务3——进入保护模式

任务要求

复现“进入保护模式”一节，使用gdb或其他debug工具在进入保护模式的4个重要步骤上设置断点，并结合代码、寄存器的内容等来分析这4个步骤，最后附上结果截图。gdb的使用可以参考指导网站appendix中的“debug with gdb and qemu”。

① 步骤一

首先，我们需要安装C++编译工具，安装命令如下：

安装编译工具

```
sudo apt install libssl-dev
sudo apt install libc6-dev-i386
sudo apt install gcc-multilib
sudo apt install g++-multilib
```

② 步骤二

然后，我们需要将内核编译成i386 32位版本。执行命令行如下：

将内核编译成 i386 32 位版本

```
make i386_defconfig
make menuconfig
```

(4) 实验任务4——保护模式下的汇编程序设计

在进入保护模式后，按照如下要求，编写并执行一个自己定义的32位汇编程序，要求简单说一说你的实现思路，并提供结果截图。

任务要求

使用两种不同的自定义颜色和一个自定义的起始位置(x, y)，使得bootloader加载后，在显示屏坐标(x, y)处开始输出自己的学号+姓名拼音首字母缩写，要求相邻字符前景色和背景色必须是相互对调的。

① 步骤一

首先是汇编程序的设计思路：

1) 首先，考虑到32位地址下，。。。

2) 然后，。。。

3) 最后，。。。。

结合上述设计思路，编码得到的汇编程序段如下：

代码清单6：实验任务4汇编 asm

```asm
my_function:
push ax
push bx

sub ax, 10
sub bx, 10
; 后进先出
pop ax
ret
```

② 步骤二

接着，通过执行下面的命令编译并运行汇编程序：

编译并利用 QEMU 运行程序

```
nasm -o mbr.o -g -f elf32 mbr.asm
ld -o mbr.symbol -melf_i386 -N mbr.o -Ttext 0x7c00
qemu-system-i386 -hda hd.img -s -S -parallel stdio -serial null
```

执行汇编程序后，可得到图3所示的结果：

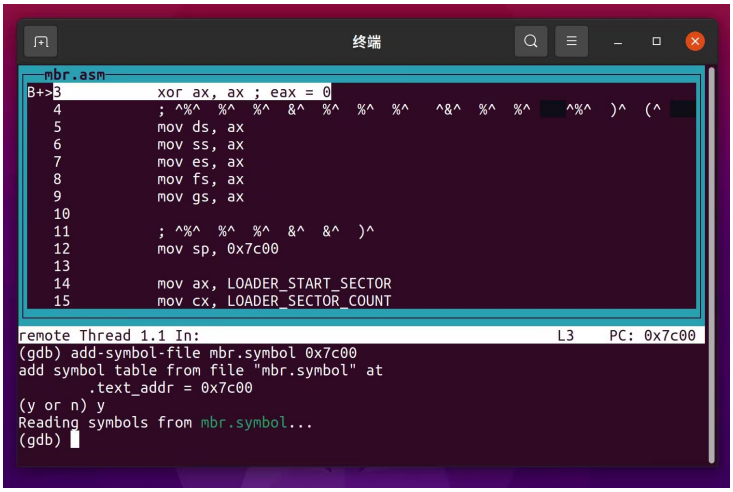


图3 实验结果

五、 实验总结与心得

通过本次实验，我解决了中大学子在撰写实验报告时的排版问题，减轻了排版工作量，排版结果更标准、更专业.....

六、 附录：实验代码清单

实验任务1的C++代码如下：

代码清单7: 实验任务1

```cpp
#include <iostream>
#include <vector>
using namespace std;
int main(){
    // 累加 20 以内的素数
    int sum = 0;
    for(int e : {2, 3, 5, 7, 11, 13, 17, 19}){ // 用 auto 类型更合理
        sum += e;
    }
    cout << sum << endl;
    // 输出结果 77
    int arr[] = {1, 3, 5, 7, 9};
    // 声明数组 arr,初始化为 5 个奇数
    for(auto &ele : arr){
        // 声明 ele,与数组 arr 关联在一起,用了 auto
        ele = ele * 2;
        // 修改数组每个元素的值
        cout << ele << " ";
        // 输出 ele,2 6 10 14 18
    }
    cout << endl;

    for(auto ele : arr){
        cout << ele << " ";
    }
    // 没有改变:1 3 5 7 9
    cout << endl;
    return 0;
}
```

实验任务2的汇编代码如下：

代码清单8: 实验任务2

```asm
org 0x7c00
[bits 16]
```

```
xor ax, ax
; 初始化段寄存器
mov ds, ax
mov ss, ax
mov es, ax
mov fs, ax
mov gs, ax

call read_disk
jmp 0x0000:0x7e00

jmp $

read_disk:
        mov ax, 0
        mov es, ax
        mov bx, 0x7e00
        mov ah, 0x02
        mov al, 0x05
times 510-($-$$) db 0
db 0x55, 0xaa
```

七、 参考文献及资料

• x86汇编语言——从实模式到保护模式. 李忠等. 电子工业出版社

• 第三章-从实模式到保护模式 https://gitee.com/nelsoncheung/sysu-2023-spring-operating-system/tree/main/lab3

• LBA向CHS模式的转换 https://blog.csdn.net/G_Spider/article/details/6906184