

操作系统实习

谢迪

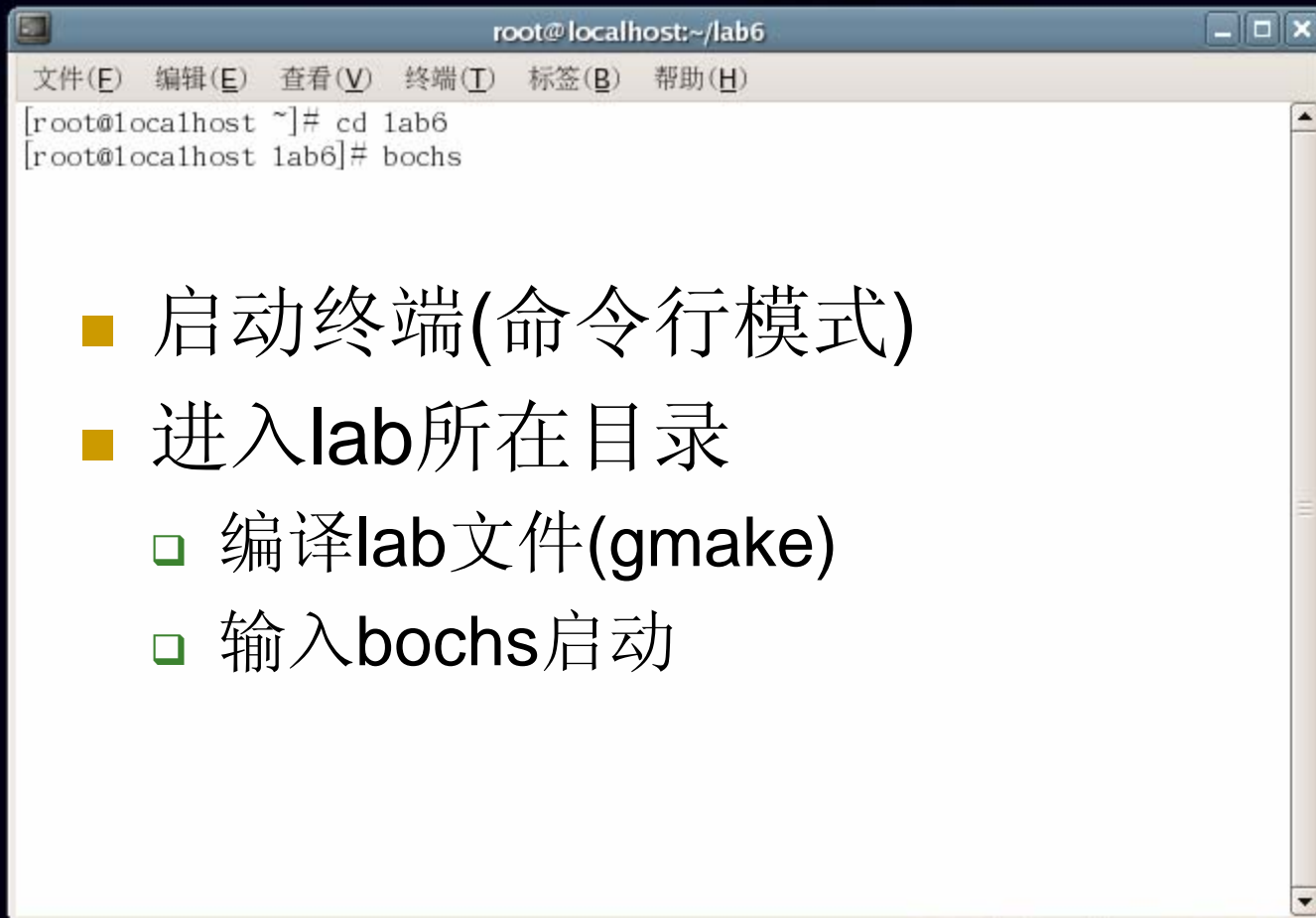
系统安装

- 使用VmWare虚拟机
 - 快捷：不用安装，方便切换回Windows
 - 但对物理机器性能要求稍高，特别是内存
- 也可以直接在Windows下安装Bochs(不推荐)

系统安装 - Bochs

- 参考<http://pdos.csail.mit.edu/6.828/2005/tools.html>
 - ❑ 在命令行下输入:
 - ❑ `tar xzvf bochs-2.2.1.tar.gz`
 - ❑ `cd bochs-2.2.1`
 - ❑ `./configure --enable-disasm \`
 - ❑ `--enable-debugger \`
 - ❑ `--enable-new-pit \`
 - ❑ `--enable-all-optimizations \`
 - ❑ `--enable-4meg-pages \`
 - ❑ `--enable-global-pages \`
 - ❑ `--enable-pae \`
 - ❑ `--enable-all-optimizations \`
 - ❑ `--disable-reset-on-triple-fault \`
 - ❑ `--with-all-libs`
 - ❑ `make`
 - ❑ `make install`

Bochs的使用

A screenshot of a terminal window titled 'root@localhost:~/lab6'. The window has a menu bar with '文件(E)', '编辑(E)', '查看(V)', '终端(T)', '标签(B)', and '帮助(H)'. The terminal shows the following commands and output:

```
[root@localhost ~]# cd lab6
[root@localhost lab6]# bochs
```

- 启动终端(命令行模式)
- 进入lab所在目录
 - 编译lab文件(gmake)
 - 输入bochs启动

Fedora

Bochs的使用



Bochs的使用

■ 基本操作

□ C

- 继续执行

□ S

- 执行一行

□ Ctrl + C

- 停止Bochs执行，这时才可以进行调试和执行其他Bochs命令

□ q

- 退出Bochs

Bochs的使用

类型	命令	功能
调试	vb addr	设置一个虚拟地址上的断点
	lb addr	设置一个线性地址上的断点
	b(pb) addr	设置一个物理地址上的断点
运行	s	单步或者连续多次单步执行
	c	继续执行
查看内存	x(xv) /nuf addr	查看给出虚拟地址的内存情况 其中参数n为显示的单位数 u为每单位的大小 f是显示的格式
	xp /nuf addr	查看给出物理地址的内存情况 参数意义同上
查看寄存器	info r	显示CPU的整数寄存器和内容
	info eflags	显示解码后的EFLAGS寄存器内容
查看GDT内容	info gdt[a[b]]	显示gdt表的内容，其中a是显示的起始表项，b是终止表项

代码合并

■ 获取原始代码

- ❑ `mkdir given-code`
- ❑ `cd given-code`
- ❑ `tar xzf ../lab1.tar.gz`
- ❑ `cd ..`

- ❑ `mv given-code/lab1 lab1-unchanged`

■ 比较原始代码与自己修改后的代码，获取差异部分放到lab1-changes.txt中

- ❑ `diff -r -u lab1-unchanged lab1 > lab1-changes.txt`

■ 让程序自动将修改部分更新到新的lab代码中

- ❑ `cd lab2`

- ❑ `patch -p1 -u < ../lab1-changes.txt`

代码合并

- 失败则需手工修改更新
 - 可能有多个文件修改失败，需要逐个处理
 - 一个文件中所有修改点的更新状态都会给出：
 - Hunk #xx FAILED at xxx
 - Hunk #xx succeeded at xxx
 - 还会写出：
 - x out of xx hunks FAILED – saving rejects to file [filename]
 - 则需要打开给出的rej文件，手工比对修改。

*** 232,238 ***

```
//      - pages -- kernel RW, user NONE
//      - the image mapped at UPAGES -- kernel R, user R
// Your code goes here:
```

```
// Check that the initial page directory has been set up correctly.
check_boot_pgdir();
```

--- 257,264 ---

```
//      - pages -- kernel RW, user NONE
//      - the image mapped at UPAGES -- kernel R, user R
// Your code goes here:
```

```
+ pages = boot_alloc(npage * sizeof(struct Page), PGSIZE);
+ boot_map_segment(pgdir, UPAGES, npage * sizeof(struct Page), PADDR(pages),
PTE_U | PTE_P);
```

```
// Check that the initial page directory has been set up correctly.
check_boot_pgdir();
```

*** 309,315 ***

```
n = ROUNDUP(npage*sizeof(struct Page), PGSIZE);
for (i = 0; i < n; i += PGSIZE)
    assert(check_va2pa(pgdir, UPAGES + i) == PADDR(pages) + i);
```

心得

- 根据实验要求，多阅读**Lab**的源代码
 - 很多要填的内容跟已经提供的代码功能相似，可以通过阅读来获得启示
 - 阅读时需要从大处着眼，先了解每个函数的功能，它们的调用关系，怎么实现要完成的功能
- 利用**grade**来调试
 - 出问题后，察看**grade.sh**文件，找出测试点，根据测试点再查找自己的相关代码
 - 源代码中包含一部分给评分程序使用的代码，可以根据它们来完成代码和找问题