OS

Theory

Lab

前言

2025秋开设的《操作系统》课程共设置6次实验课，涵盖5个实验项目，总计24个学时。前四个实验项目均以开源操作系统xv6为基础，但已根据哈尔滨工业大学（深圳）的课程要求进行了针对性的修改和扩展，各占4个学时。第五个实验分配8个学时，由我校学生自主设计的文件系统实验案例。

- 实验指导书：https://os-labs.p.cs-lab.top
- 实验代码下载地址：https://gitee.com/ftutorials/xv6-oslab24-hitsz.git

xv6是MIT于2006年开发的一个用于教学的操作系统，它是在x86架构上用ANSI标准C重新实现的Unix第六版（即v6），课程编号为6.828。2019年xv6被移植到RISC-V架构之上，并设置了新的课程编号6.S081。

- MIT 6.S081：https://pdos.csail.mit.edu/6.828/2020/index.html
- xv6 book：https://pdos.csail.mit.edu/6.828/2020/xv6/book-riscv-rev1.pdf

部署实验环境

> **Tip**
>
> 若你使用的是Windows操作系统，推荐利用WSL2来安装并运行Linux环境，当然也可以通过虚拟机来安装Linux。
>
> WSL2(Windows Subsystem for Linux 2)：WSL2不仅能够在Windows中高效运行Linux内核，而且相比于WSL1具有更好的性能和完全的Linux系统调用兼容性。WSL2支持运行主流的Linux发行版（Ubuntu、Debian等），并能够直接访问Windows的文件系统，可以通过Windows自带的文件资源管理器，像操作Windows的文件那样操作WSL上的文件，并且可以在WSL和Windows之间直接进行文件拖拽。通过WSL2可以避免使用虚拟机，占用的系统资源更少，操作也更加便捷。

1.所需工具

- **Linux发行版**：Linux发行版是由Linux内核、GNU工具、附加软件和软件包管理器组成的操作系统。所谓“发行”，是由某些机构“发行”了Linux内核、所有必要的软件及实用程序，使其可以作为一个操作系统使用。主流的Linux发行版有Ubuntu、Debian、Fedora、Centos等，国产Linux发行版有openEuler、Deepin等。
- **RISC-V工具链**：包括一系列交叉编译的工具，如gcc、binutils、glibc等，用于把源码编译成机器码。
- **QEMU模拟器**：用于在x86架构的电脑上模拟RISC-V架构的CPU。在实验中，我们将通过QEMU模拟器观察xv6的运行过程，QEMU通过模拟取指、译码、执行等步骤来仿真RISC-V处理器的操作。

2.具体步骤

1. 下载binutils-gdb.zip并将其上传到Linux环境中。
2. 更新软件源，确保后续安装的工具均为最新版本。
   `sudo apt-get update`
3. 安装基础开发工具：git、编译工具、调试工具、QEMU模拟器、RISC-V交叉编译链。
   `sudo apt-get install git build-essential gdb-multiarch qemu-system-misc gcc-riscv64-linux-gnu binutils-riscv64-linux-gnu`
4. 安装编译binutils-gdb依赖包。

`sudo apt-get install autoconf automake autotools-dev curl python3 python3-pip libmpc-dev libmpfr-dev libgmp-dev gawk build-essential bison flex texinfo gperf libtool patchutils bc zlib1g-dev libexpat-dev ninja-build git cmake libglib2.0-dev libslirp-dev python3-dev`

5. 解压工具包并进入目录。

```
# 以工具包在~/Downloads为例，若路径不同请替换
unzip ~/Downloads/binutils-gdb.zip -d ~/
cd ~/binutils-gdb*    # *通配符匹配带哈希值的长目录名
```

6. 创建并进入编译目录（避免污染源码）。

`mkdir build && cd build`

7. 配置编译选项。

```
# --target: 指定目标架构为RISC-V 64位
# --prefix: 指定安装路径为/usr/local
# --with-python: 关联Python3支持
../configure --target=riscv64-unknown-elf --prefix=/usr/local --with-python=/usr/bin/python3
```

8. 编译与安装（耗时约5-10分钟，-j4表示4核并行编译）。

```
make -j4
sudo make install
```

9. 安装RISC-V架构交叉编译GCC。

```
sudo apt-get update
sudo apt-get install gcc-riscv64-unknown-elf
```

10. 进入用户主目录。

`cd ~`

11. 克隆实验仓库。

`git clone https://gitee.com/ftutorials/xv6-oslab24-hitsz.git`

12. 进入实验目录并切换分支。

```
cd xv6-oslab24-hitsz
git checkout util
```

13. 启动xv6。
    `make qemu`
14. 终端输出末尾显示以下内容，即表示xv6启动成功。

```
init: starting sh
$
```

15. 退出xv6的方法，务必按照以下步骤退出以避免进程残留。
    - 先按`Ctrl + a`
    - 松开后立刻按`x`

一、Linux开发环境基础知识

1.常用文件管理命令

1.1 Linux文件管理

Linux文件路径的描述本质上是一棵树，目录和文件就是树中的各种结点。

根目录`/`：根目录是文件系统这棵树的根结点。

家目录`~`：是特定用户的私人工作区域，每个用户登录系统后，默认的起始位置就是该用户的家目录。可以使用`echo $HOME`命令来查看当前用户的家目录。

- 普通用户的家目录位于`/home/用户名`
- root用户的家目录位于`/root`

通过`cd /`命令进入根目录，输入`ls`并回车，可以看到以下文件夹：

- `bin`：存储系统启动、维护和修复所必需的可执行程序（二进制文件或命令）
- `etc`：代理服务器的配置文件。
- `var`：里面会有log（日志）
- `lib`：安装包和库文件（静态链接库）等
- `home`：用户的家目录
- `proc`：进程相关的信息

1.2 Linux路径描述

通过`cd ~`命令回到家目录，然后执行以下命令：

```
mkdir tmp
touch tmp/main.cpp
```

Linux中有两种方式描述这个`main.cpp`文件的路径：

- **绝对路径**：从根目录开始描述路径——`/home/zsh/tmp/main.cpp`。
  - 绝对路径的描述必须以根目录`/`为开头。
- **相对路径**：从当前目录开始描述路径——`tmp/main.cpp`。

我们可以随时通过`pwd`命令来查看当前路径，其输出是当前命令的绝对路径。

在路径描述里，Linux有3个特殊目录：

- `.`：表示当前目录。执行`cd .`不会改变当前路径。
- `..`：表示上一级目录。
- `~`：表示家目录。

1.3 常用快捷键

- `Ctrl+c`：用于中止当前正在运行的进程。
- `Ctrl+u`：清空本行命令。
- `Tab键`：可以补全命令和文件名，若补全不了则快速按两下`Tab`键，可以显示备选选项。
- `方向键↑`：可以调出之前执行过的指令。

1.4 常用命令

- `pwd`：显示当前工作目录的绝对路径
- `touch <filename>`：创建一个空文件或更新现有文件的时间戳。
- `cat <filename>`：查看文件内容，将文件内容输出到标准输出（通常是终端）。

- `cd`：更改当前工作目录。
  - `cd 目录路径`：切换到指定目录路径。
  - `cd`：不加路径则返回家目录。

    - `cd -`：返回上一个待过的路径。
    - `cd ..`：返回上层目录。
- `cp`：复制文件或目录。
    - `cp file1 file2`：将文件1复制到同一目录下并命名为文件2。
    - `cp -r a b`：将目录a及其内容递归复制到目录b中（如果b不存在，则会被创建）。
- `mkdir`：创建新目录。
    - `mkdir 目录名`：创建单个目录。
    - `mkdir g\ s`：创建包含空格的目录（如“g s”），需使用反斜杠\进行转义。
    - `mkdir -p a/b/c`：一次性创建多级目录（若a或b不存在，则一并创建）。
- `ls`：列出当前目录下的所有文件，蓝色的是文件夹，白色的是普通文件，绿色的是可执行文件（具体颜色可能因系统配置而异）。
    - `-l`：以列表形式展示详细信息，包括文件权限、所有者、大小等。
    - `-hl`：以人类易读的形式显示文件大小，使文件大小更易于阅读（如以KB、MB为单位）。
    - `-a`：查询所有文件（包括隐藏文件，所有以`.`开头的文件都是隐藏文件）。
- `rm`：删除文件或目录。
    - `rm <filename>`：删除指定的文件。
    - `rm -r 目录名`：递归删除整个目录及其内容。
    - `rm -f <filename>`：强制删除文件，不提示确认。
- `mv`：移动/重命名文件或目录。
    - `mv <filename> 目录`：将文件移动到对应目录里。
    - `mv a.txt b.txt`：将 `a.txt` 重命名为 `b.txt`。

2.开发常用命令

> **Tip**
>
> 如果不会使用某个命令，可以用`man 某命令`或`某命令 --help`来查看该命令的使用说明。

2.1 系统状况

- `top`：查看所有进程的信息（Linux的任务管理器）。
  - 打开后，输入 `M`：按使用内存排序。
  - 打开后，输入 `P`：按使用CPU排序。
  - 打开后，输入 `q`：退出。
- `df -h`：查看硬盘使用情况。
- `free -h`：查看内存使用情况。
- `du -sh`：查看当前目录占用的硬盘空间。
- `ps aux`：查看所有进程。
- `kill -<SIGNAL> <pid>`：采用信号`SIGNAL`对应的方式结束编号为编号为`pid`的进程。
  - `kill -9 <pid>`：杀死编号为`pid`的进程（9是信号`SIGKILL`的编号）。
  - `kill -15 <pid>`：正常结束编号为`pid`的进程（15是信号`SIGTERM`的编号）。
  - 某些程序必须要正常结束而不能杀死，因此关闭某个程序前最好先查阅文档。
- `netstat -nt` ：查看所有网络连接。
- `w` ：列出当前登陆的用户。
- `ping www.baidu.com`：检查是否连网。
  - Linux上的`ping`命令是不会自动终止的，需要使用`Ctrl+c`手动终止。

2.2 文件权限

- 文件的权限表示有10bit。bit1表示该文件是文件夹，还是超链接（快捷方式）。bit2-10按序三位一组，每组取值0-7，表示信息如下表：

组权限	bit序号（2-10）	可读权限	可写权限	可执行权限
所有者权限	bit2-4	bit2	bit3	bit4
同组用户权限	bit5-7	bit5	bit6	bit7
其他用户权限	bit8-10	bit8	bit9	bit10

- `chmod`：修改文件权限。
  - `chmod +x <filename>`：给文件添加可执行权限。
    - 如何执行这个文件程序？使用命令`./<filepath>` （如果文件在当前目录下，则 `./<filename>`）。

    - `chmod -x <filename>`：去掉文件的可执行权限。
    - `chmod 777 <filename>`：将文件的权限改成777（即比特串111_111_111，代表最高权限）。
    - `chmod 777 目录名 -R` ：递归修改整个文件夹的权限。
- 举例：在终端输入`ls -l`，可以看到所有文件的权限。假设在家目录下有一个脚本文件`main.sh`，它的权限是`-rw-r--r--`，则这串比特串的含义如下：
    - bit1是`-`，表示该文件是普通文件。
    - bit2-4是`rw-`，表示该文件对于所有者来说，可以读和写，但不能执行。
    - bit5-7是`r--`，表示该文件对于同组用户来说，只读，不能写和执行。
    - bit8-10是`r--`，表示该文件对于其他用户来说，只读，不能写和执行。
    - 如果执行`chmod +x main.sh`，则bit4会变为`x`，表示这个文件可以执行。
    - 如果要在家目录下执行`main.sh`，使用`./main.sh`命令即可。

2.3 文件检索

- `find 目录路径 -name '*.c'`：搜索某个目录下所有后缀为`.c`的文件。
- `grep xxx`：从标准输入(`stdin`)中读入若干行数据，如果某行中包含`xxx`，则输出该行，否则忽略该行。
    - `grep xxx 文件名或正则表达式`：查找名称匹配文件名或正则表达式的文件里，有没有`xxx`字符串。
- `wc`：统计行数、单词数、字节数。
    - 既可以从 `stdin` 中直接读入内容；也可以在命令行参数中传入文件名列表。
    - `wc <filename>`：按序输出行数、单词数、字节数。
    - `wc -l`：统计行数。
    - `wc -w`：统计单词数。
    - `wc -c`：统计字节数。
- `cut`：分割一行内容。
    - 从 `stdin` 中读入多行数据。
    - `echo $PATH | cut -d ':' -f 3,5`：输出 `PATH` 用 `:` 分割后第3、5列数据。
    - `echo $PATH | cut -d ':' -f 3-5`：输出 `PATH` 用 `:` 分割后第3-5列数据。
    - `echo $PATH | cut -c 3,5`：输出 `PATH` 的第3和第5个字符。
    - `echo $PATH | cut -c 3-5`：输出 `PATH` 的第3-5个字符。
- `sort`：将每行内容都按照字典序排序。

    - 可以从 `stdin` 中读取多行数据。
    - 可以从命令行参数中读取文件名列表。
- `xargs`：将`stdin`中的数据用空格或回车分割成命令行参数。
    - `find . -name '*.c' | xargs cat | wc -l`：统计当前目录下所有C语言文件的总行数。
    - 如果不加 `xargs` 也会有输出，但是这个输出统计的不是 `.c` 文件。
    - 不加 `xargs` ， `cat` 输出的就是 `stdin` 文件的内容，统计的也就是 `stdin` 的行数。

2.4 查看文件内容

- `head -3 <filename>`：展示文件的前3行内容，同时也支持从`stdin`读入内容。
- `tail -3 <filename>`：展示文件的末3行内容，同时也支持从`stdin`读入内容。
- `more <filename>`：浏览文件内容。
    - 回车：下一行。
    - 空格：下一页。
    - `b`：上一页。
    - `q`：退出。
- `less <filename>`：与`more`类似但功能更齐全。
    - 回车：下一行。
    - `y`：上一行。
    - `Page Down`：下一页。
    - `Page Up`：上一页。
    - `q`：退出。

2.5 用户相关操作

- `history`：展示当前用户的历史操作，内容存放在 `~/.bash_history` 中。
    - `history` 展示的历史操作包含当前bash界面用户使用过的最近2000条操作，而 `.bash_history` 里的历史操作不包含当前未关闭的bash里的操作。只有关闭bash之后，才会把操作写入 `.bash_history` 文件。

2.6 工具

- `time 命令`：统计命令的执行时间。
- `watch -n 0.1 命令`：每0.1秒执行一次命令。
- `diff <filename1> <filename2>`：查找文件1和文件2的不同之处。
- `tar`：压缩与解压缩文件。
  - `tar -zcvf 压缩文件名.tar.gz 目录路径/*`：将目标文件夹下所有文件压缩为`.tar.gz`类型文件。
  - `tar -zxvf 压缩文件名.tar.gz`：解压缩。
- `md5sum <filename>`：计算文件的`md5`哈希值。
  - 计算任意长度的字符串的哈希值。
  - 可以从`stdin`读入内容，也可以在命令行参数中传入文件名列表。
  - **用途**：
    - 计算文件的哈希值，以核对文件是否准确且完好无损。
    - 加密信息，比如加密密码。数据库存储密码时都不会存储原密码字符串，而是存储密码的哈希值。用户输入密码后，先转化成哈希值再比对。因为哈希值几乎不可能反推，所以如果忘记密码就只能重设密码，而不能找回密码。

2.7 查看计算机信息

- `cat /etc/os-release`：查看系统信息。
- `lscpu`：查看CPU信息。
- `free`：查看内存信息。
- `fdisk -l`：查看磁盘信息
- `date`：显示当前日期和本地时间。
- `date --utc/--u`：显示当前日期和UTC时间。

3.Linux管道和重定向

3.1 管道命令要点

管道(Pipeline)是Unix/Linux系统中的一个强大特性，它允许将一个命令的标准输出`stdout`直接作为另一个命令的标准输入`stdin`，从而实现命令之间的数据传递和连续处理。

1. 管道命令仅处理`stdout`，忽略标准错误输出`stderr`。
2. 管道右侧的命令必须能够接受`stdin`。
3. 多个管道命令可以串联使用。

3.2 管道命令举例

举例：`find . -name '*.c' | xargs cat | wc -l`

- `find -name '.*c'`：在当前目录及其子目录下寻找所有C语言文件，并输出每个目标文件的路径。
- `xargs`：读取`find`命令的输出，并将这些文件路径作为参数传递给`cat`命令，此处使用`xargs`是为了处理文件路径中可能包含特殊字符或空格的情况。
- `cat`：读取`xargs`传递的文件路径，并将每个路径对应的文件内容输出到`stdout`，成为下一个命令`wc`的输入。
- `wc -l`：读取`cat`命令的输出（即所有C语言文件的内容），并统计这些内容的总行数，最终输出结果。

3.3 重定向

在Unix/Linux系统中，重定向允许用户将命令的输入、输出或错误输出重定向到某个文件或其他命令中。

重定向是Shell的功能，而非程序自带的功能。程序只是与文件描述符打交道，Shell负责将文件描述符与文件或设备关联起来。

基本概念	文件描述符	功能	终端关联设备
标准输入`stdin`	0	Unix/Linux程序默认从`stdin`读取数据	连接到键盘
标准输出`stdout`	1	Unix/Linux程序默认向`stdout`输出数据	显示在屏幕上
标准错误`stderr`	2	Unix/Linux程序默认向`stderr`流写入错误信息	显示在屏幕上，但与`stdout`分开

> ⚠ **Warning**
>
> 重定向符`>`、`>>`、`2>`、`2>>`、`>&`、`&>`、`<`与文件描述符之间不能有空格！但与文件名之间可以有空格，尽管为了清晰起见也可能不加空格。

P1 输出重定向

- 标准输出重定向：
  - `command >file`：将命令的标准输出重定向到file中，如果file已存在则覆盖其内容，否则创建该file。
  - `command >>file`：将命令的标准输出以追加的方式重定向到file中，如果file已存在则在file末尾追加内容，否则创建该file。
- 标准错误重定向：
  - `command 2>file`：将命令的标准错误重定向到file中。
  - `command 2>>file`：将命令的标准错误以追加的方式重定向到file中。
- 标准输出和标准错误同时重定向：
  - `command >file 2>&1`：首先将命令的标准输出重定向到file中，然后将标准错误重定向到标准输出当前指向的位置（即file）。此种方法在旧版Shell中常用。
  - `command &> file`：这是现代Shell（如bash）提供的一种更简洁的语法，可以同时重定向标准输出和标准错误到file中。
- 特殊重定向：
  - 当重定向到`/dev/null`时，表示丢弃输出。`/dev/null`是一个特殊的设备文件，向它写入的数据会被系统丢弃，读取它则立即返回EOF(End Of File)。

P2 输入重定向

- `command <file`：将命令的标准输入重定向到file中，这样命令就会从file中读取输入数据，而非从键盘读取。

P3 其他重定向

- 使用文件描述符进行重定向：
  - `n >file`：将文件描述符n的输出重定向到file中。这里n可以是`0(stdin)`、`1(stdout)`、`2(stderr)`**之外**的任何文件描述符，但通常用于自定义的文件描述符。
  - `n >>file`：将文件描述符n的输出以追加的方式重定向到file中。

- `n >&m`：将文件描述符n的输出重定向到文件描述符m的当前位置。这通常用于合并不同来源的输出。
- `n <&m`：将文件描述符n的输入重定向到文件描述符m的当前位置。这较少使用，因为输入重定向通常直接通过`<`符号完成。

- Here Document（内嵌文档）：
  - `<<delimiter`：将开始标记delimiter和结束标记delimiter之间的内容作为输入传递给命令。这种方式允许直接在命令行中提供多行输入。

4.环境变量
