实验报告：Pintos 安装

课程名称：操作系统	年级：2023 级本科	上机实践成绩：
指导教师：张民	姓名：张梓卫	
上机实践名称：Pintos 安装	学号：10235101526	上机实践日期：
上机实践编号：（1）	组号：	上机实践时间：

目录

一 实验目的

本次实验为操作系统的第一次实验，目的是安装并运行 Pintos 操作系统，并熟悉操作系统的基本概念、原理和实践方法。实验中，还对当下流行的 Docker 容器技术进行了实践，基于 PKU-OS 进行教学，让我们能够顺便熟悉 Linux 操作系统中的部分命令，为以后的开发打下基础。

二 内容与设计思想

利用 Docker 在 Windows 11 上安装基于 x86 的 PintOS（斯坦福大学）。

三 使用环境

使用 Docker v27.1.1 进行 Pintos 的安装实验，基于 Windows 11 操作系统使用 WSL2。
实验报告使用 LaTeX 进行撰写，使用 Vim 编辑器进行文本编辑。

四 实验过程与分析

1 使用 Docker 拉取镜像

```
docker run -it pkuflyingpig/pintos bash
```

拉取镜像

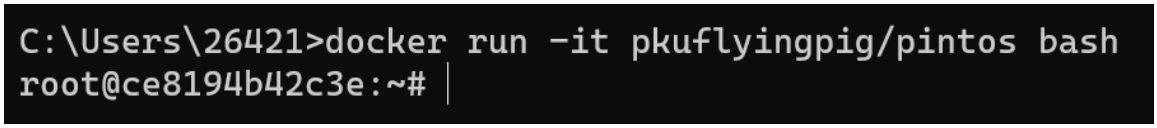

图 1: 拉取镜像后进入 Docker 容器

2 从 Github 克隆代码

注意，文档里写的这个无法拉取，需要使用 SSH 协议，需要在 GitHub 上设置 SSH 密钥。

```
> git clone hit@github.com:PKU-OS/pintos.git
```

克隆代码

所以使用了助教发送的新地址：

```
> git clone https://gitee.com/duerwuyi/pintos.git
```

克隆代码

Clone 代码结果如下：

图 2: 克隆代码

其中，Docker 中的 Log 显示了克隆的进度

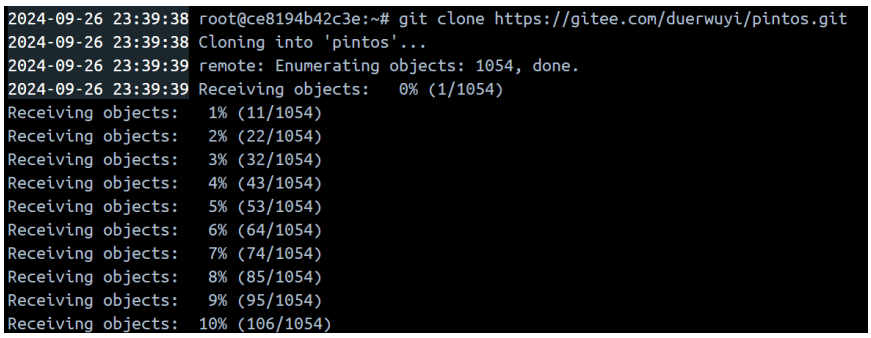

图 3: 克隆进度

3 使用 cd、pwd、ls 命令等进入 threads

首先，对 cd、pwd、ls 等命令进行复习：

3 .1 cd 命令

- cd 进入用户主目录；
- cd ~ 进入用户主目录；
- cd - 返回进入此目录之前所在的目录；
- cd .. 返回上级目录（若当前目录为“/”，则执行完后还在“/”；“..”为上级目录的意思）；
- cd ../.. 返回上两级目录；
- cd !$ 把上个命令的参数作为 cd 参数使用。

3 .2 ls 命令

- -a：列出目录下的所有文件，包括以. 开头的隐藏文件；
- -l：除了文件名之外，还将文件的权限、所有者、文件大小等信息详细列出来。

3 .3 pwd 命令

全称 Print Working Directory，显示当前工作目录的完整路径。

4 进入 threads 目录

```
> cd pintos
> ls
> cd src
> ls
> cd threads
```

进入 threads 目录

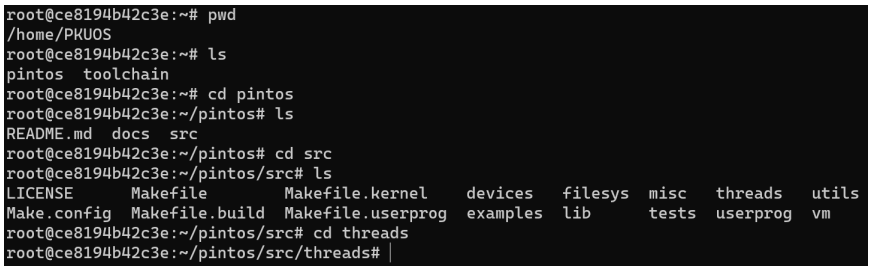

图 4: 进入 threads 目录

这部分疯狂使用 `ls` 命令是为了在命令行中没有界面时能够及时获取当前文件夹的信息。

5 编译 Pintos

在当前目录使用 make 指令，即可编译 Pintos。

```
> make
```

编译 Pintos

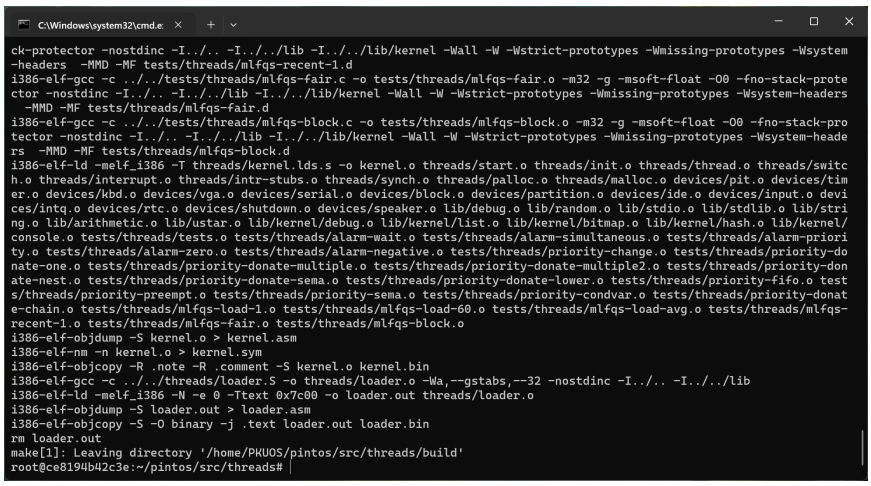

图 5: 编译 Pintos

通过观察可以发现，输出结果中提示，此时已经离开了 build 目录。

6 运行 Pintos

```
> pintos —
```

运行 Pintos

输入该命令，输出的提示信息如下，代表 Boot 已完成，Pintos 已成功编译。

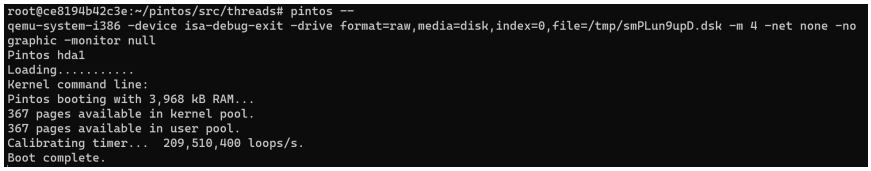

图 6: 运行 Pintos

7 完成 alarm-multiple

7 .1 进入 build 文件夹

当完全 Boot 完成后，会发现终端卡在了 Boot Complete, 而没有退出回到可以交互的状态。
此时,使用 Ctrl + C 组合键打断此时的状态,然后回退到可交互的 Bash 终端中,可以发现,回退到了 `/pintos/src/thread`
通过 `ls` 指令查看当前文件夹，并且通过 `cd build` 进入 build 文件夹内

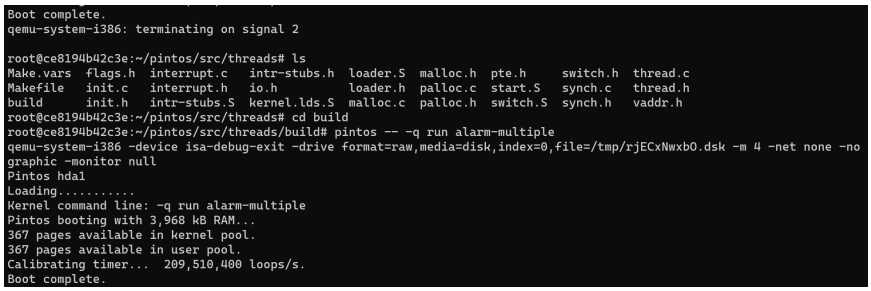

图 7: 进入 build 文件夹

7 .2 Run alarm-multiple

```
> pintos -- -q run alarm-multiple
```

Run alarm-multiple

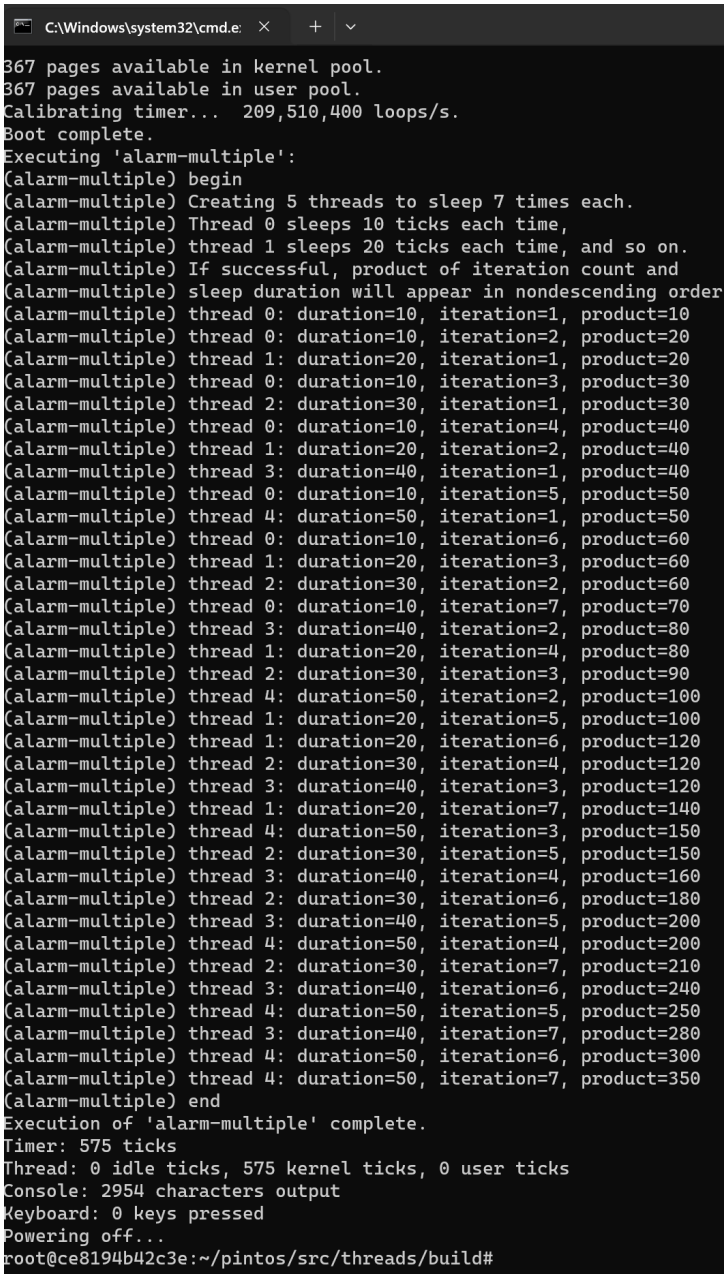

图 8: 运行 alarm-multiple

8 实验结束，进行后续预热

在 Docker 中的 Container 里运行时，交互的命令不是 Windows，使用的是 Exit 命令来退出当前的 Container。

- **退出交互模式**:
  - 使用 ‘exit‘ 命令：退出当前的交互终端并关闭容器（如果容器是通过交互模式启动的）。
  - 使用快捷键 ‘Ctrl + D‘：在没有挂起的进程时，按下 ‘Ctrl + D‘ 也会退出交互模式。
- **保持容器运行并退出交互模式**:
  - 使用快捷键 ‘Ctrl + P‘ 然后 ‘Ctrl + Q‘：容器继续在后台运行，你可以从交互模式中分离。
    通过以下命令重新连接到容器：

```
docker attach <container_id_or_name>
```

- **从宿主系统停止容器**:
  - 列出所有正在运行的容器：docker ps
  - 使用 ‘docker stop‘ 命令停止容器：

```
docker stop <container_id_or_name>
```

- **在容器内部运行命令关闭**:
  - 使用 ‘shutdown‘ 命令关闭容器内的操作系统：shutdown now
  - 使用 ‘halt‘ 命令停止当前容器系统：

五 实验结果总结

通过在虚拟机中运行的方式，无法将本地文件与 Container 内的文件相关联，于是我选择在主机上 Clone 代码，然后挂载到 Docker 中运行。

首先，Clone 代码到本地（Path: C:
Users
26421
pintos）

使用如下代码将 Pintos 目录挂载到 Docker 中，以后所有的变化都会同步到本地。

```
docker run -it --rm --name pintos --mount type=bind,source=C:\Users\26421\pintos, target=/home/PKUOS/pintos
  pkuflyingpig/pintos bash
```

这一部分是参考官方文档：https://pkuflyingpig.gitbook.io/pintos/getting-started/environment-setup

因为我们所有的文件更改都会同步到 Windows 系统中，所以要加入 -rm 参数告诉 Docker 每次运行结束后就删除 Container，不然会占用很多的存储空间。

Now when you `ls`, you will find there is a new directory called `pintos` under your home directory in the container, **it is shared by the container and your host computer i.e. any changes in one will be synchronized to the other instantaneously**.

现在当你 `ls` 时，你会发现在容器的主目录下有一个名为 `pintos` 的新目录，它被容器和你的主机共享，即其中一个的任何更改都会立即同步到另一个。

图 9: 挂载目录

1 自定义 Windows 命令实现 Alias

由于每一次我们都需要在 Windows Bash 终端中输入同样的命令以挂载到 Docker 中，所以我们可以自定义 Windows 命令，以省略每一次输入大量命令的步骤。

新建批处理文件

新建 PintosPintos.txt 于 Windows Desktop 中，输入以下命令：

```
@echo off
docker run -it --rm --name pintos --mount type=bind,source=C:\Users\26421\pintos, target=/home/PKUOS/pintos
  pkuflyingpig/pintos bash
```

PintosPintos.txt

其中，@echo off 是用来隐藏批处理文件执行时的命令行输出，只显示实际的执行结果，使脚本运行时的输出更加简洁明了。

之后，将这个文本文件重命名为 Pintos.bat，当桌面目录添加至系统的环境变量 PATH 中，于是使用 Win + R，输入 cmd，再输入 Pintos.bat 即可运行 Pintos 容器。

六 附录

本实验参考了以下资源：

- PKU-OS 文档：https://pkuflyingpig.gitbook.io/pintos/getting-started/environment-setup
- Docker 官方文档：https://docs.docker.com/