xv6环境配置

2020年秋季麻省理工学院(MIT)提供了两门单独的课程：6.828和6.S081。其中，6.S081是为本科生开设一门独立的AUS（Advanced Undergraduate Subjects）课程，主要讲授操作系统入门知识。6.828是为研究生级别提供研讨式课程，重点研究操作系统。

6.828？官方文档：https://pdos.csail.mit.edu/6.828/2020/index.html

6.S081？官方文档：https://pdos.csail.mit.edu/6.828/2020/tools.html

Win10子系统安装：https://docs.microsoft.com/zh-cn/windows/wsl/install-win10

实验环境主要包括：RISC-V工具链、QEMU模拟器、xv6三个部分。

**RISC-V工具链**：包括一系列交叉编译的工具，用于把源码编译成机器码，如gcc，binutils，glibc等。

**QEMU模拟器**：用于在我们机器上(X86)模拟RISC-V架构的CPU。

**xv6**：MIT实验教学系统使用xv6，它是一个简化的类Unix系统。xv6是Dennis Ritchie和Ken Thompson开发的Unix Version 6（v6）操作系统的再实现。它在一定程度上遵守v6的结构和风格，但它是用ANSI C实现的。xv6早期版本是基于x86多核处理器实现的，最新版本基于多核RISC-V实现。RISC-V是开源指令集架构，可以免费地用于所有设备中。任何人可以基于RISC-V指令集架构设计服务器CPU、家用电器CPU、工控CPU以及传感器中所用的CPU。

实验环境：Windows 10 + Ubuntu 18.04LTS（子系统）

**一、安装RISC-V工具链**

1. 直接克隆MIT提供的RISC-V GNU编译器工具链的存储库。

（注：如果没有安装git软件，请先安装：sudo apt-get install git）

git clone --recursive https://github.com/riscv/riscv-gnu-toolchain

（可能出现的问题1）

或者

git clone https://github.com/riscv/riscv-gnu-toolchain

2. 克隆完成后进入riscv-gnu-toolchain目录

cd riscv-gnu-toolchain

git submodule update --init --recursive

3. 接下来，为了确保您拥有编译工具链所需的软件包，执行下列命令

sudo apt-get install autoconf automake autotools-dev curl libmpc-dev libmpfr-dev libgmp-dev gawk build-essential bison flex texinfo gperf libtool patchutils bc zlib1g-dev libexpat-dev

（可能出现的问题2）

4. 编译riscv-gnu-toolchain

cd riscv-gnu-toolchain

./configure --prefix=/usr/local

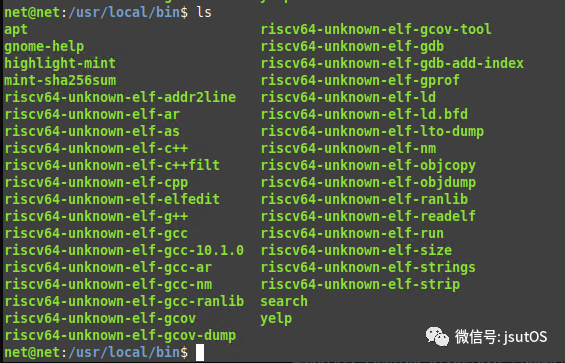
sudo make

（可能出现的问题3：

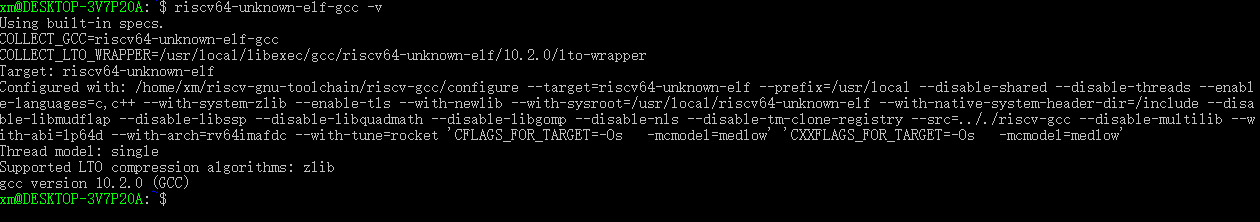
）

cd ..

查看/usr/local/bin/目录如下图所示。



此时在命令行输入riscv64-unknown-elf-gcc -v，如果能显示版本信息则代表安装成功。



(2) QEMU模拟器

下载qemu模拟处理器，我们编译生成的risc-v平台的机器码，需要通过模拟cpu执行。

接下来，检索并提取qemu4.1的源代码。

$ wget https://download.qemu.org/qemu-4.1.0.tar.xz

#下载后解压并进入目录

$ tar xf qemu-4.1.0.tar.xz

为riscv64-softmmu构建QEMU

$ cd qemu-4.1.0

$ ./configure --disable-kvm --disable-werror --prefix=/usr/local --target-list="riscv64-softmmu"

=======================================

安装失败：



如果上述配置失败，请安装下列两个软件包：

$ sudo apt-get install libglib2.0-dev

$ sudo apt-get install libpixman-1-dev

然后重新配置。

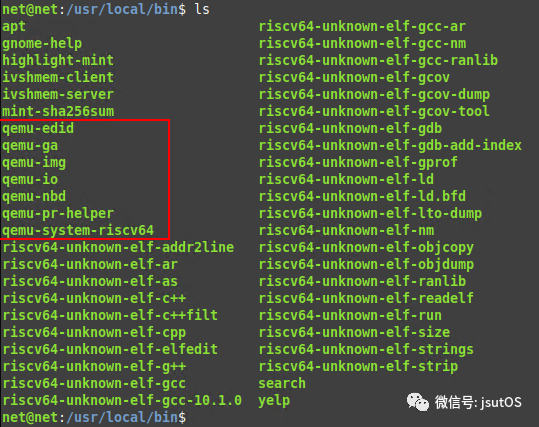
$ ./configure --disable-kvm --disable-werror --prefix=/usr/local --target-list="riscv64-softmmu"

=======================================

$ make

$ sudo make install

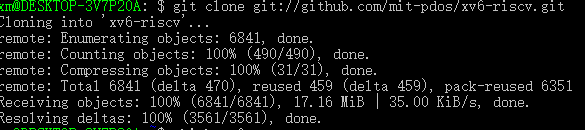
查看/usr/local/bin/目录如下图所示。



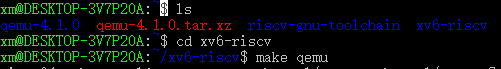
三、xv6源码

从github下载xv6源码

git clone git://github.com/mit-pdos/xv6-riscv.git

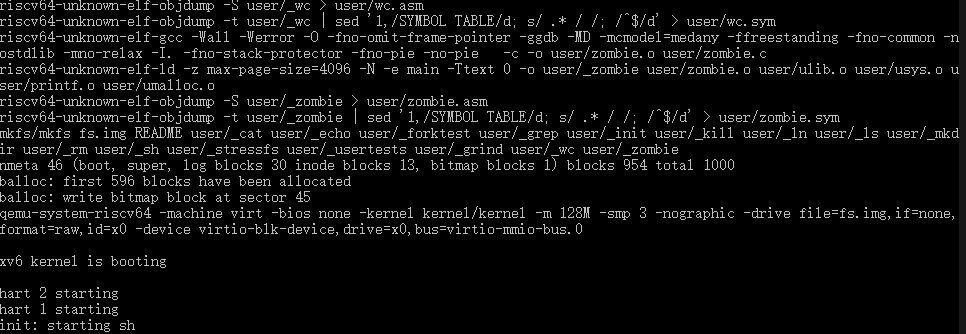


进入xv6目录下进行编译，并启动qemu仿真运行：

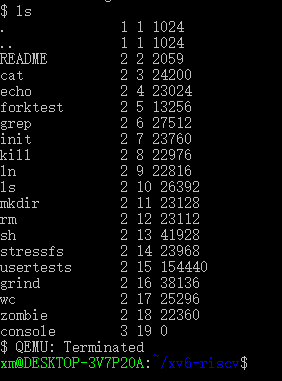


make qemu

运行结果如下图：



在xv6命令行下，输入ls，看到正确回应，那么恭喜你，至此运行环境已经设置成功。



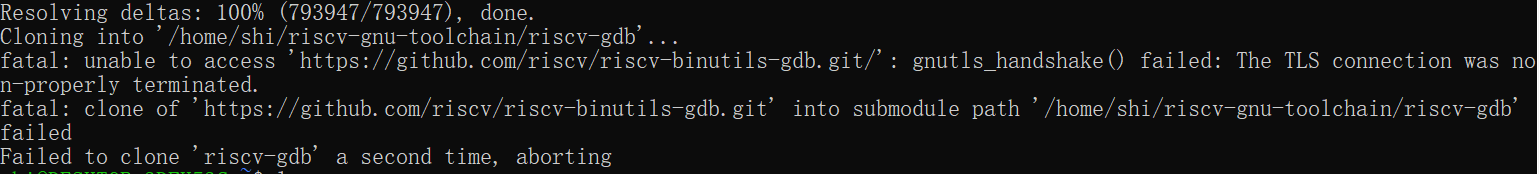
如果要退出qemu, 按下Ctrl 键和A键, 然后释放这两个键，再按X键即可退出

6.S081中文翻译网址：[简介 - MIT6.S081 (gitbook.io)](https://mit-public-courses-cn-translatio.gitbook.io/mit6-s081/)

[Lec01 Introduction and Examples (Robert) - MIT6.S081 (gitbook.io)](https://mit-public-courses-cn-translatio.gitbook.io/mit6-s081/lec01-introduction-and-examples)

**可能出现的问题**

1. 部分文件克隆失败

****

解决方法：

（1）下载riscv gun工具链，网络环境不好的化，可以按下面的方法，一个模块一个模块单独下载：

git clone https://github.com/riscv/riscv-gnu-toolchain

cd riscv-gnu-toolchain

git clone --recursive https://github.com/riscv/riscv-qemu.git

git clone --recursive https://github.com/riscv/riscv-newlib.git

git clone --recursive https://github.com/riscv/riscv-binutils-gdb.git

git clone --recursive https://github.com/riscv/riscv-dejagnu.git

git clone --recursive https://github.com/riscv/riscv-glibc.git

git clone --recursive https://github.com/riscv/riscv-gcc.git

（2）这样下载的目录和前一种方式不同，主要是qemu和gdb,binutils三个模块，可以进行一些copy工作，使得qemu, riscv-binutils,riscv-gdb三个目录有数据，否则编译会出错。

cd riscv-qemu

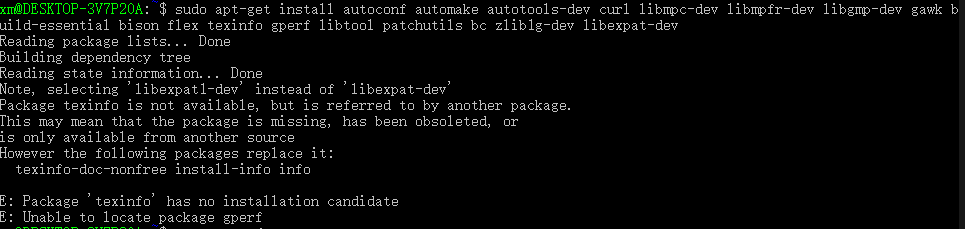
cp -a \* ../qemu

cd riscv-binutils-gdb

cp -a \* ../riscv-gdb

cp -a \* ../riscv-binutils

**2.**  E：Package ‘texinfo’ has no installation candidate



问题描述：package “packagename” has no installion candidate

出错原因为没有更新apt的下载源

没有更新apt的下载源，这个问题比较简单，网上也有很多这样问题的解决办法：

#sudo apt-get update

#sudo apt-get upgrade

#sudo apt-get install <packagename>

**3. 编译不成功**



1. 设置环境变量

vim ~/.bashrc

export RISCV="/usr/local" #工具的安装链接

export PATH=$PATH:$RISCV/bin #该路径下为链接工具

:wq #保存退出

source ~/.bashrc

1. 编译riscv-gnu-toolchain

cd riscv-gnu-toolchain #克隆完成进入克隆下来的目录

mkdir build;

cd build #创建并进入build文件夹

../configure --prefix=$RISCV --enable-multilib #检查环境并生成当前环境使用的Makefile

sudo make #注意是make而不是make linux, make编译得到是我们最常用的unknown-elf-gcc等工具链，而make linux则是unknown-linux-gnu-gcc,整个make大概需要一个小时