Installing the xv6 kernel on Mac (Intel/M1/M2)

This document outlines the steps to install the MIT-version of the xv6 kernel on MacOS, particularly on the M1/M2 SoC. The following steps are also tested on MacBook with x86 (Intel chip).

1 xv6

1.1 Update the virtual/physical Linux OS

Log in to your (virtual or physical) Linux operating system (*e.g.*, Ubuntu). Update and upgrade packages.

```
> sudo apt update
> sudo apt upgrade -y
```

Listing 1. Update and upgrade packages (sudo not required by default for OS running on Docker)

1.2 Installing tools

Install the following tools.

```
> sudo apt install \
    git \
    wget \
    curl \
    vim \
    qemu \
    build-essential \
    gdb-multiarch \
    qemu-system-misc \
    gcc-riscv64-linux-gnu \ # or gcc-x86-64-linux-gnu
    binutils-riscv64-linux-gnu # or binutils-x86-64-linux-gnu
```

Listing 2. Installing essential tools

Installation of the last few tools may fail if attempting to install on incompatible architecture. Figure out the appropriate version for your hardware architecture from Ubuntu Packages.

1.3 Download the appropriate xv6 version

MIT CS AI Lab's public version of the xv6 kernel seems to work on x86 and x86-64 architectures (*i.e.*, 32- and 64-bit Intel-based x86 architecutres). The RISCV version seems to work particularly for ARM-based architectures (Apple M1/M1 SoC, for example).

```
# for ARM architectures (Apple M1/M2)
> git clone https://github.com/mit-pdos/xv6-riscv
> cd xv6-riscv

```

```
# for Intel-based x86 architectures
> git clone https://github.com/mit-pdos/xv6-public
> cd xv6-public
```

Listing 3. Download xv6

If needed, make the following changes in `Makefile`.

```
QEMU = qemu-system-riscv64 // or qemu-system-x86_64
```

Listing 4. Configure `Makefile`

1.4 Build the kernel

```
> make
> make qemu # or make qemu-nox
> ls # lists some commands upon successful installation
```

Listing 5. Build and open the kernel

Note

1. To compile your custom program files, include the filename in the variable list `UPROGS` in `Makefile`
2. To include non-code files (like text files), include the filename in the `fs.img` recipe in `Makefile` similar to `README`
3. xv6-riscv is organised slightly differently from xv6-public. For xv6-riscv, including header files is nested into the folders `kernel` or `user`. For example:

   ```
   #include "kernel/types.h"
   #include "kernel/stat.h"
   #include "user/user.h"

   ...
   ```

   Listing 6. Including header files for xv6-riscv

   It is recommended that you place your custom program files in the `user` folder.

   Also, some minor syntax variations exist (*e.g.*, , `printf`, `exit`, *etc.*) which become apparent upon compilation

2 Docker for OS virtualisation

Docker is an OS-virtualisation platform. If you wish to avoid installing other third-party virtual machine clients, you may choose to use Docker instead. Docker is lightweight and works on UNIX-based operating systems, MacOS, and (I think) Windows, too. The following is purely CLI-based.

2.1 Installing Docker

Install Docker with Homebrew.

```
> brew install docker
```

Listing 7. Install Docker with Homebrew

After installing Docker and each time you wish to use it, make sure to start it (or set it to automatically restart every time the system is booted). Docker may be started either from the GUI or through the CLI.

```
> open -a docker
```

Listing 8. Start Docker

2.2 Downloading Ubuntu

Download a copy (a.k.a. image) of Ubuntu from Docker Hub.

```
> docker pull ubuntu
```

Listing 9. Download Ubuntu from Docker Hub

Refer to Docker commands or cheat sheets to know about more Docker commands.

2.3 Running Ubuntu

An instance (a.k.a. container) of the Ubuntu image may be started with the following command:

```
> docker run -it -d --name <container_name> ubuntu /bin/bash
```

Listing 10. Start an instance of Ubuntu. `<container_name>` may be any string to name the instance (*e.g.*, `OS6611Ubuntu`)

2.4 Logging into the Ubuntu instance

The steps so far need be done only once or if setting up for the first time. To log in to the Ubuntu instance each time, run the following command:

```
> docker start -i <container_name>
```

Listing 11. Log in to Ubuntu. `<container_name>` is the name of the instance from the previous step

Upon successfully logging in, you should be placed in the root folder of the Ubuntu instance. To verify, run any of the following commands:

```
> uname -a # OS/user/CPU architecture versions
> cat /etc/os-release # OS details
> lscpu # CPU details
```

Listing 12. Verifying login

If appropriate details are printed, the OS is installed, started, and logged into successfully.