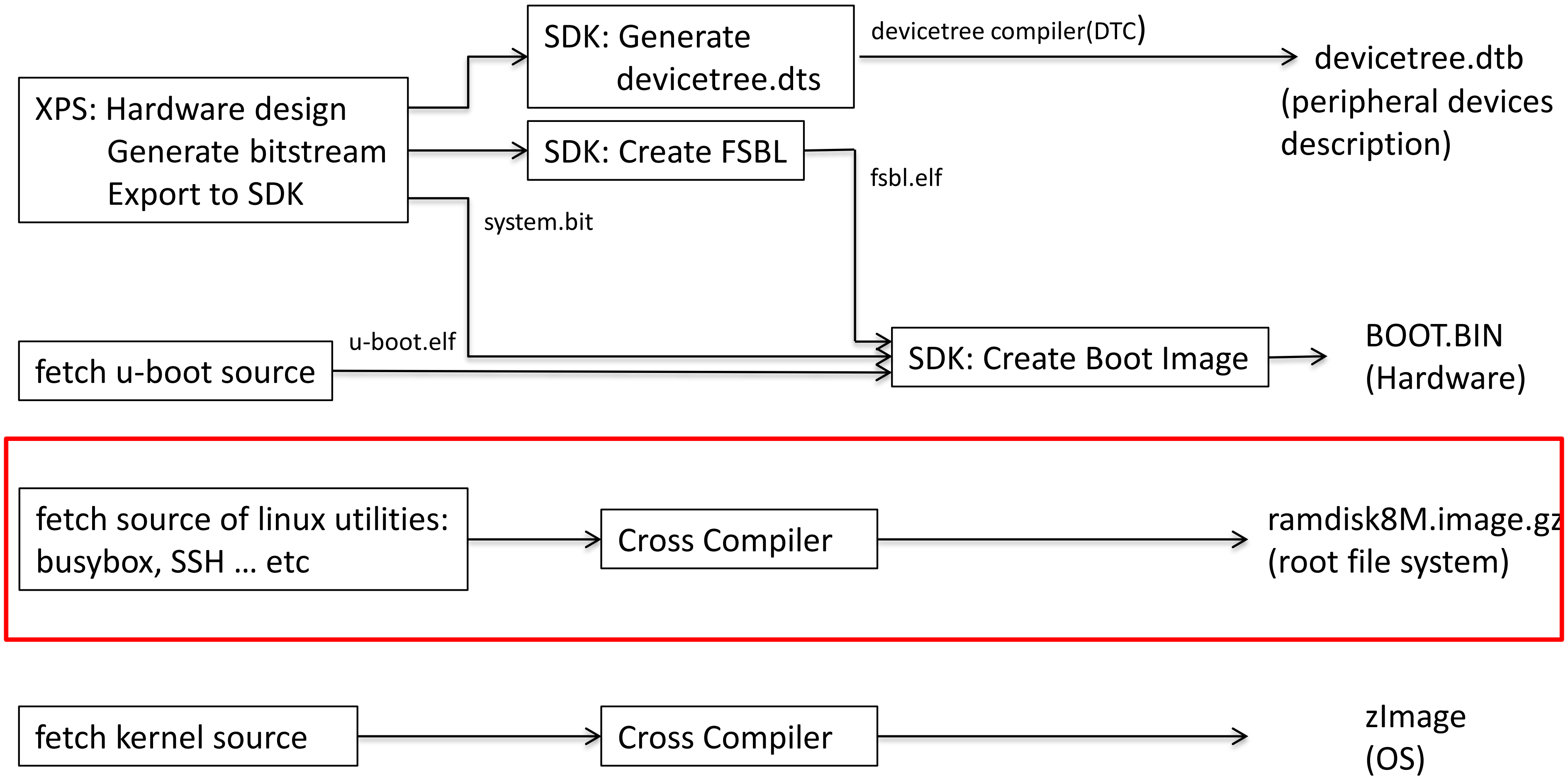
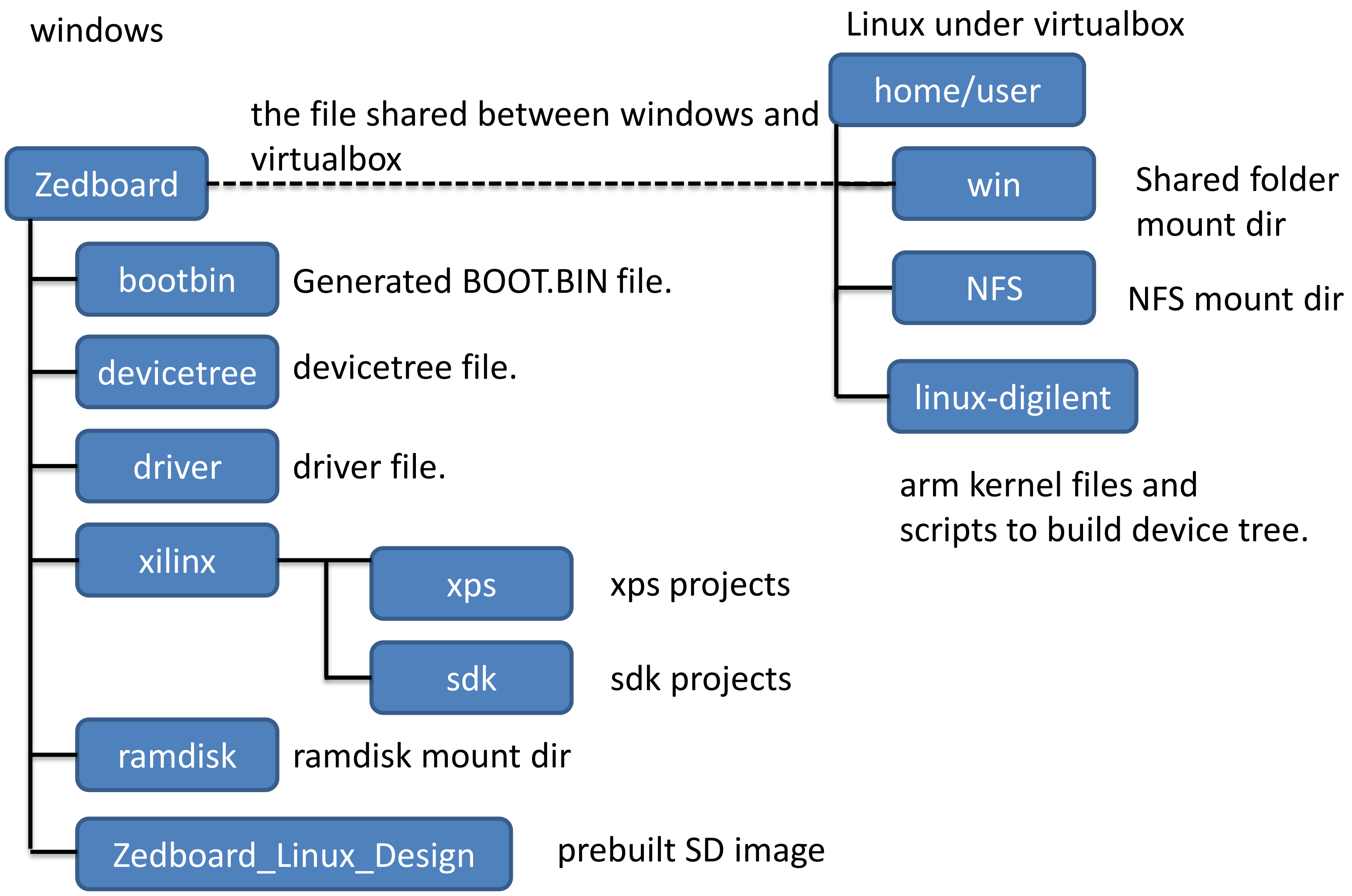
ZedBoard Lab 3
Ramdisk

Chun-Chen Tu
timtu@umich.edu


XPS: Hardware design
Generate bitstream
Export to SDK
SDK: Generate devicetree.dts
devicetree compiler(DTC)
devicetree.dtb (peripheral devices description)
SDK: Create FSBL
fsbl.elf
system.bit
u-boot.elf
fetch u-boot source
SDK: Create Boot Image
BOOT.BIN (Hardware)
fetch source of linux utilities: busybox, SSH … etc
Cross Compiler
ramdisk8M.image.gz
(root file system)
fetch kernel source
Cross Compiler
zImage
(OS)

File placement

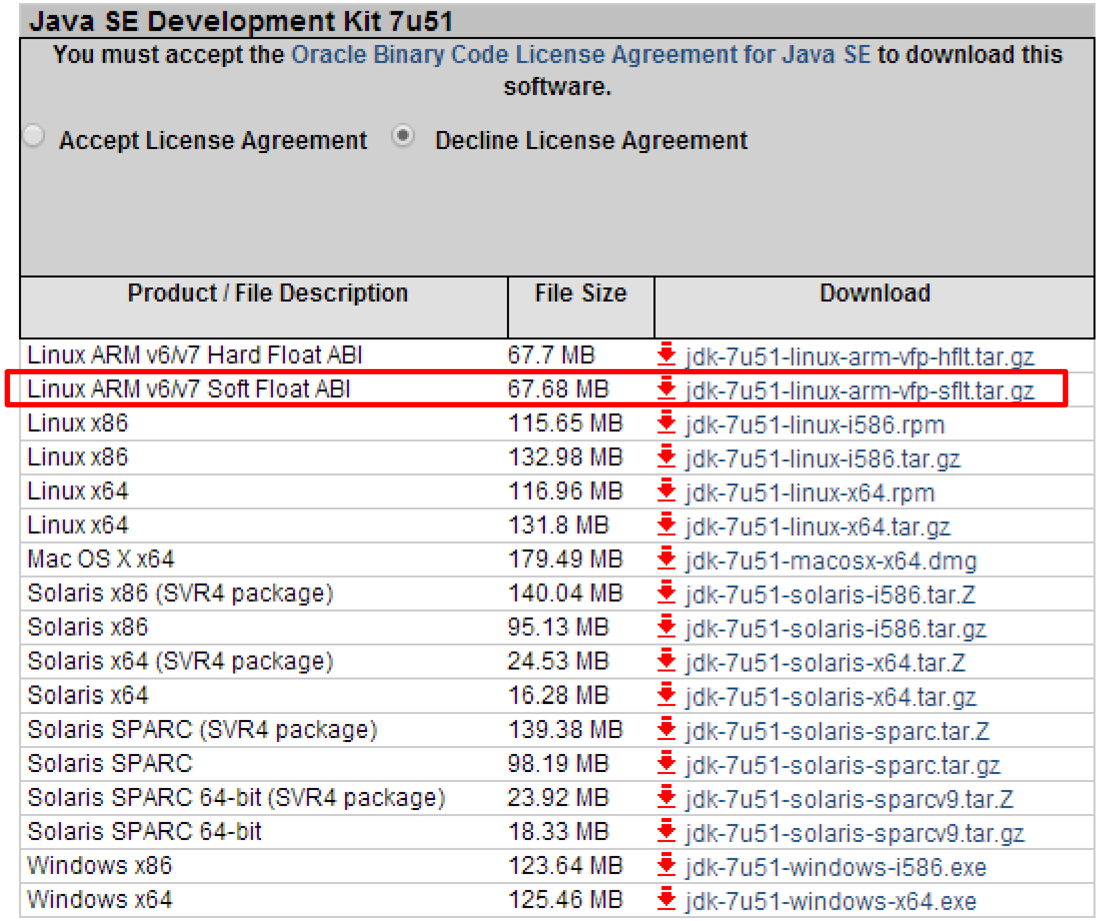


Ramdisk

- Actually, it contains root file system(RTFS), in a format of ramdisk.
  - When booting up, OS will create a ramdisk and load root file system into it.
- The goal in this slide is to modify ramdisk so it will support standalone Hadoop.
  - There will be another slide explaining about distributed Hadoop.

Java on ARM

- Install Java: got to Java downalod JDK7 website
  - Download the soft float version



Java SE Development Kit 7u51

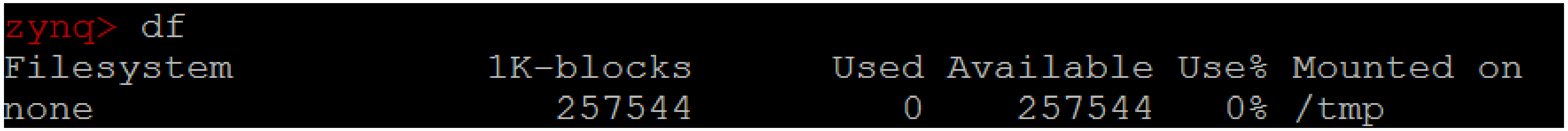
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement [✓ Decline License Agreement]

Product / File Description	File Size	Download
Linux ARM v6/v7 Hard Float ABI	67.7 MB	jdk-7u51-linux-arm-vfp-hflt.tar.gz
Linux ARM v6/v7 Soft Float ABI	67.68 MB	jdk-7u51-linux-arm-vfp-sflt.tar.gz
Linux x86	115.65 MB	jdk-7u51-linux-i586.rpm
Linux x86	132.98 MB	jdk-7u51-linux-i586.tar.gz
Linux x64	116.96 MB	jdk-7u51-linux-x64.rpm
Linux x64	131.8 MB	jdk-7u51-linux-x64.tar.gz
Mac OS X x64	179.49 MB	jdk-7u51-macosx-x64.dmg
Solaris x86 (SVR4 package)	140.04 MB	jdk-7u51-solaris-i586.tar.Z
Solaris x86	95.13 MB	jdk-7u51-solaris-i586.tar.gz
Solaris x64 (SVR4 package)	24.53 MB	jdk-7u51-solaris-x64.tar.Z
Solaris x64	16.28 MB	jdk-7u51-solaris-x64.tar.gz
Solaris SPARC (SVR4 package)	139.38 MB	jdk-7u51-solaris-sparc.tar.Z
Solaris SPARC	98.19 MB	jdk-7u51-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	23.92 MB	jdk-7u51-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	18.33 MB	jdk-7u51-solaris-sparcv9.tar.gz
Windows x86	123.64 MB	jdk-7u51-windows-i586.exe
Windows x64	125.46 MB	jdk-7u51-windows-x64.exe

Zedboard internal storage

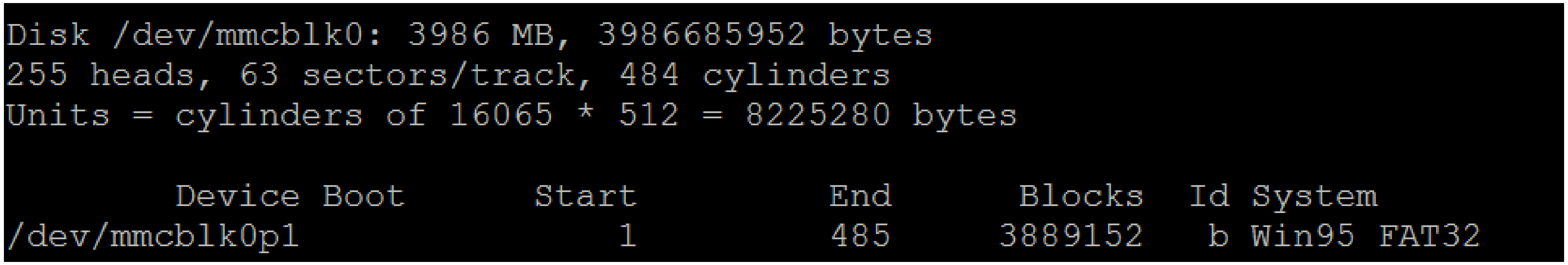
- The internal memory is only 32MB, not enough for JAVA files.(try “df” and you can see the internal memory size)



```
zynq> df
Filesystem           1K-blocks      Used Available Use% Mounted on
none                    257544         0    257544   0% /tmp
```

- One solution is to put JAVA on SD card. Put the downloaded files into SD card and boot up.

  fdisk –l

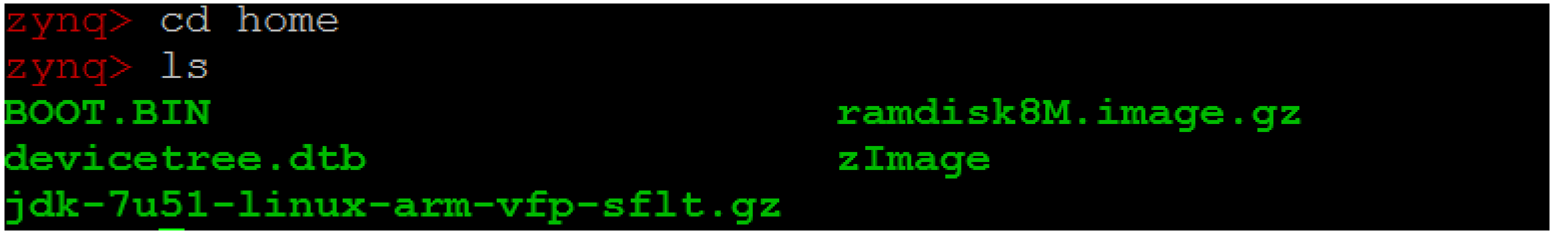


```
Disk /dev/mmcblk0: 3986 MB, 3986685952 bytes
255 heads, 63 sectors/track, 484 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

        Device Boot      Start         End      Blocks  Id System
/dev/mmcblk0p1               1         485     3889152   b Win95 FAT32
```

mkdir /home

mount /dev/mmcblk0p1 /home

cd /home



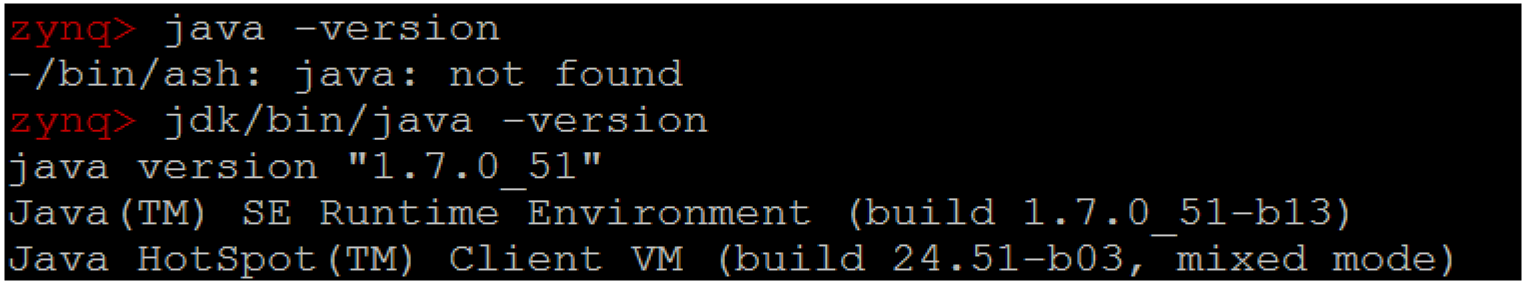
```
zynq> cd home
zynq> ls
BOOT.BIN                                ramdisk8M.image.gz
devicetree.dtb                          zImage
jdk-7u51-linux-arm-vfp-sflt.gz
```

tar xvf jdk-7u51-linux-arm-vfp-sflt.gz

mv jdk1.7.0_51 jdk (optional but will make things easy)

java -version

jdk/bin/java –version



```
zynq> java -version
-/bin/ash: java: not found
zynq> jdk/bin/java -version
java version "1.7.0_51"
Java(TM) SE Runtime Environment (build 1.7.0_51-b13)
Java HotSpot(TM) Client VM (build 24.51-b03, mixed mode)
```

Need to set environment variable to make “java” recognizable.

(Create a file and source it)

vi bashrc

export JAVA_HOME=/home/jdk

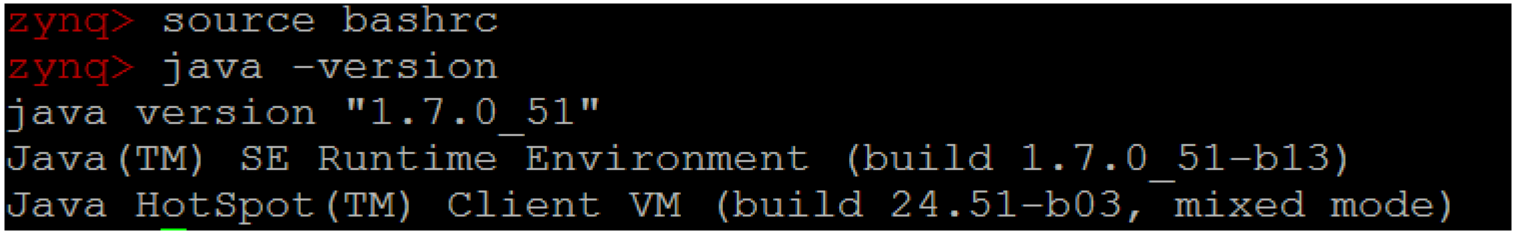
export JRE_HOME=/home/jdk/jre

export PATH=$JAVA_HOME/bin:$JRE_HOME/bin:$PATH

export CLASSPATH=$CLASSPATH:.:$JAVA_HOME/lib:$JAVA_HOME/jre/lib

source bashrc

java –version



```
zynq> source bashrc
zynq> java -version
java version "1.7.0_51"
Java(TM) SE Runtime Environment (build 1.7.0_51-b13)
Java HotSpot(TM) Client VM (build 24.51-b03, mixed mode)
```

Problems when running Hadoop

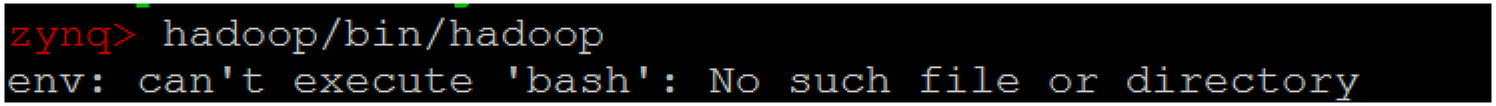
Put Hadoop files into SD card.

Boot up and mount SD card under /home just like we have done.

tar Hadoop tarball and rename the folder into hadoop.

Try to run hadoop command

hadoop/bin/hadoop

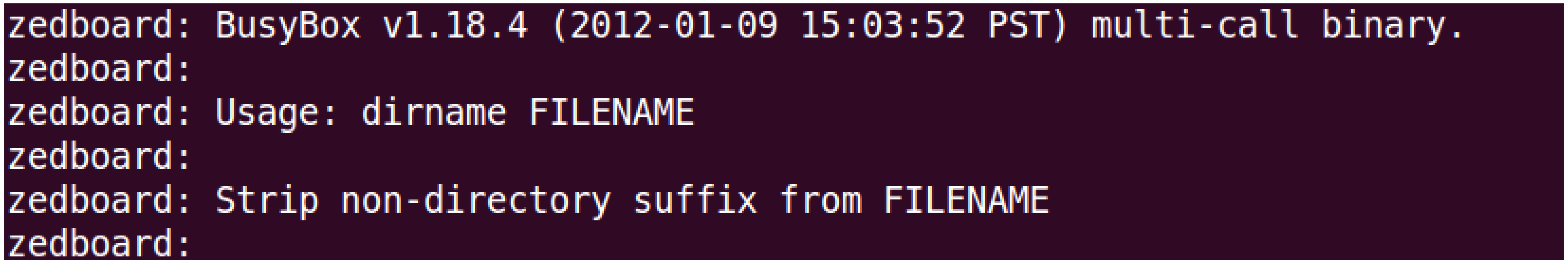


```
zynq> hadoop/bin/hadoop
env: can't execute 'bash': No such file or directory
```

No bash!!!

Other problems

- The default network setting is not correct. We should modify it.
- Hostname cannot be resolved correctly even though we add /etc/hosts files.
- No loopback interface (Only if you want to pseudo-distributed)
- "dirname" not executing well. "dirname" is one of the UNIX utilities and provided by busybox.



```
zedboard: BusyBox v1.18.4 (2012-01-09 15:03:52 PST) multi-call binary.
zedboard:
zedboard: Usage: dirname FILENAME
zedboard:
zedboard: Strip non-directory suffix from FILENAME
zedboard:
```

Before fixing all of these…

- We need to know how to modify ramdisk file.
  - reference: website

In Windows: Copy the original ramdisk8M.image.gz to D:/Zedboard/ramdisk

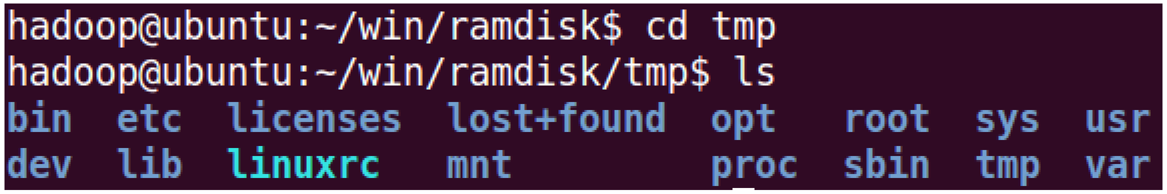
In Linux cd ~/win

gunzip ramdisk8M.image.gz

mkdir tmp

sudo mount –o loop ramdisk8M.image tmp

cd tmp



```
hadoop@ubuntu:~/win/ramdisk$ cd tmp
hadoop@ubuntu:~/win/ramdisk/tmp$ ls
bin  etc  licenses  lost+found  opt   root  sys  usr
dev  lib  linuxrc   mnt         proc  sbin  tmp  var
```

After modification

- After you make the modification, you should compress it back.

  Under Linux:

  ```
  cd ~/win/ramdisk
  sudo umount tmp
  gzip -9 ramdisk8M.image
  ```

And then you get your ramdisk8M.image.gz

Put it in SD card and boot up, you'll get your new root file system.

Outline of works

- Recompile busybox in newer version
  - Fix dirname problem. Also by doing so you can add other UNIX utilities.
  - Utilities provided by busybox may be sufficient for embedded system. For other not supported, you should fetch source code and corss compile it.
    - EX: dropbear for SSH service
- Add bash
  - Fetch bash from multistrap for arm.
  - An alternative way is to cross compile bash.
- Modify root file system
  - Network configuration: IP, routing … etc
  - Hostname resolving

Busybox

- An integrated tool for UNIX utilities. (ls, cd .. etc)
- http://www.busybox.net/downloads/
- For graphic configuration install the package

```
sudo apt-get install libncurses5-dev
```

- We download busybox-1.22.1.tar.bz2 for example

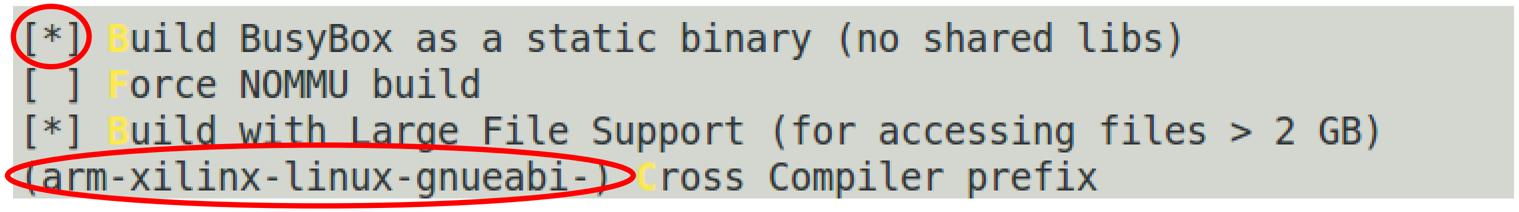
Download it

```
tar xvf busybox-1.22.1.tar.bz2
mv busybox-1.22.1 ~/busybox
cd ~/busybox
make menuconfig
```

Busybox

Busybox Settings

  Build Options



```
[*] Build BusyBox as a static binary (no shared libs)
[ ] Force NOMMU build
[*] Build with Large File Support (for accessing files > 2 GB)
(arm-xilinx-linux-gnueabi-) Cross Compiler prefix
```

(you can press "/" to search for functions your want)

remain other settings and save

make

make install

UNIX utilities shall appear under _install

All of them appears in the way of soft link to the file busybox. Thus, you can only replace the busybox file to update the root file system.(If you're not adding other utilities)

Multistrap

We need a bash for executing Hadoop.

Unfortunately, busybox doesn't provide bash.

We need to obtain bash from other ways.

Beside Multistrap, you can cross compile bash, but it's larger than Multistrap. Thus, we recommend Multistrap for space issue.

Multistrap : website

is a tool to prepare for root file system of several platforms.

NOTE: DO NOT apt-get upgrade. This will cause some problem when installing multistrap.

Prepare for multistrap

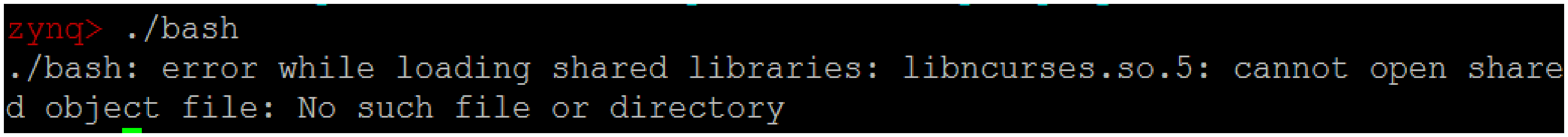
```
sudo apt-get install binfmt-support qemu qemu-user-static
sudo apt-get install multistrap

mkdir ~/multistrap
cd multistrap
mkdir rootfs
wget
ftp://hadoop:hahahadoop@140.113.114.104/multistrap.conf
sudo multistrap –f multistrap.conf
```

multistrap.conf

```
[General]
noauth=true
unpack=true
debootstrap=Debian
aptsources=Debian
arch=armel
directory=/home/hadoop/multistrap/rootfs
omitrequired=false
cleanup=true

[Debian]
packages=man-db apt sudo resolvconf ntpdate wget apt-utils coreutils
source=http://ftp.tw.debian.org/debian/
keyring=debian-archive-keyring
suite=squeeze
```

bash files

- We’re not using files from multistrap directly. Instead, we’ll copy bash to the root file system generated by busybox.
- Just copy the bash file cannot make things work. We also need to set up the related libraries properly.
- Or you’ll get the message like this:



```
zynq> ./bash
./bash: error while loading shared libraries: libncurses.so.5: cannot open share
d object file: No such file or directory
```

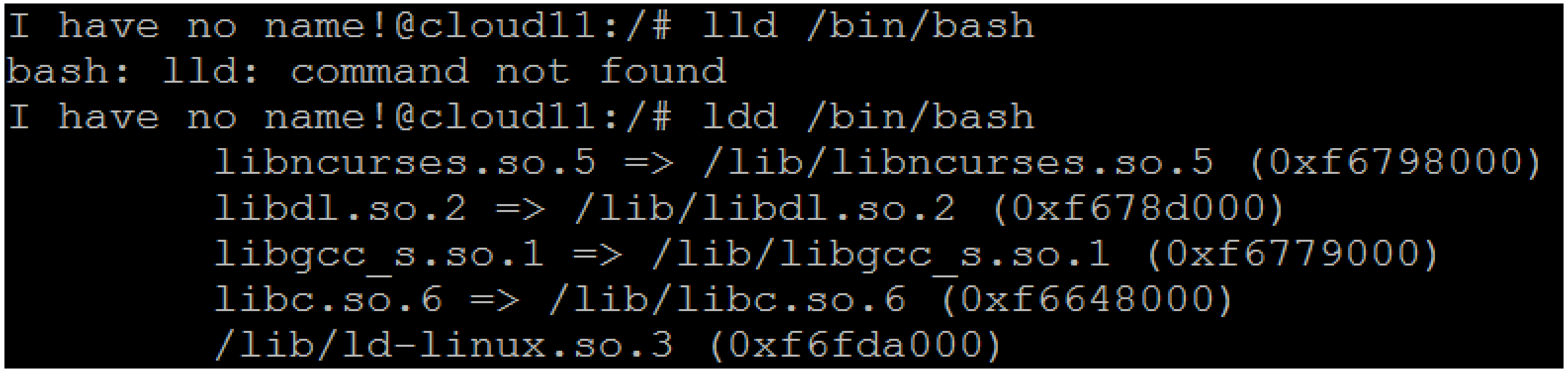
chroot and bash related libraries

- From the message we know that libncurses.so.5 is the missing library. Is there any other libraries? And how to check them?
- You can check the related libraries by *ldd* under the ARM architecture.
- We need chroot. Chroot will create an pseudo root under a specific directory. This will make you switch to ARM architecture.

```
cd ~/multistrap
sudo cp /usr/bin/qemu-arm-static rootfs/usr/bin
sudo chroot rootfs
```

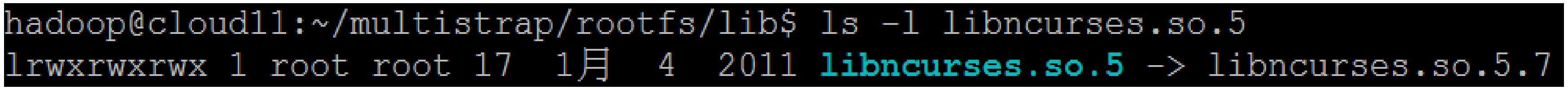
(type exit to exit chroot mode)

- Now we’re in the system of ARM architecture with root=~/multistrap/rootfs. This is called chroot jail.
- Find the bash related libraries:



```
I have no name!@cloud11:/# lld /bin/bash
bash: lld: command not found
I have no name!@cloud11:/# ldd /bin/bash
        libncurses.so.5 => /lib/libncurses.so.5 (0xf6798000)
        libdl.so.2 => /lib/libdl.so.2 (0xf678d000)
        libgcc_s.so.1 => /lib/libgcc_s.so.1 (0xf6779000)
        libc.so.6 => /lib/libc.so.6 (0xf6648000)
        /lib/ld-linux.so.3 (0xf6fda000)
```

- And then we find these related files. Fortunately, the only missing files is libncurses.so.5. Others already exist in the prebuilt root file system.



```
hadoop@cloud11:~/multistrap/rootfs/lib$ ls -l libncurses.so.5
lrwxrwxrwx 1 root root 17  1月  4  2011 libncurses.so.5 -> libncurses.so.5.7
```

- Also, we find this is a soft link to libncurses.so.5.7 . Thus, we need to copy this library into our root file system and create the soft link.

Copy busybox and bash

mount the ramdisk on ~/win/ramdisk/tmp

```
sudo cp ~/multistrap/rootfs/lib/libncurses.so.5.7
~/win/ramdisk/tmp/lib
cd ~/win/ramdisk/tmp/lib
sudo ln –s libncurses.so.5.7 libncurses.so.5
sudo cp ~/multistrap/rootfs/bin/bash ~/win/ramdisk/tmp/bin
sudo cp ~/busybox/_install/bin/busybox ~/win/ramdisk/tmp/bin
```

(NOTE: We only copy file busybox here since other commands are only soft links. However, if you add some other utilities not originally exists, you should either create soft link yourself or copy the soft link from busybox-generated root file system)

Root files system modification

- rcS: rcS is a script automatically executed when booting up.
  - We will add and modify the IP, loopback interface and routing information in this file.
- We wish to run hadoop under user hadoop. Thus we'll create an user hadoop in advance.
- For hostname resolving:
  - /etc/hostname: defining the hostname of machine
  - /etc/hosts: defining the hostname and IP mapping.
  - /etc/nsswitch.conf: the file telling system how to resolve hostname.
- We need ssh service. Dropbear is already implemented, we only need to configure for public key authentication.

rcS

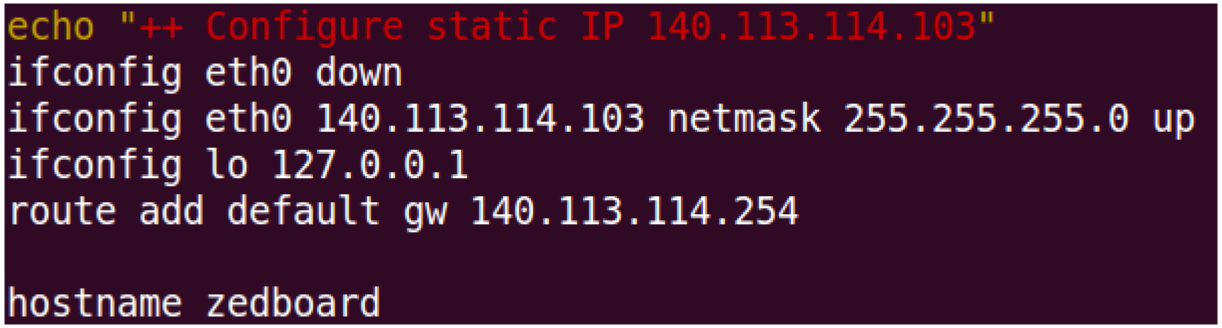
mount the ramdisk on ~/win/ramdisk/tmp

cd ~/win/ramdisk/tmp/etc/init.d

sudo vim rcS

You’ll see what have done when booting up.

- For network:
  - Modify for your IP and routing gateway.
  - Add the loopback interface.
  - Set the hostname to zedboard



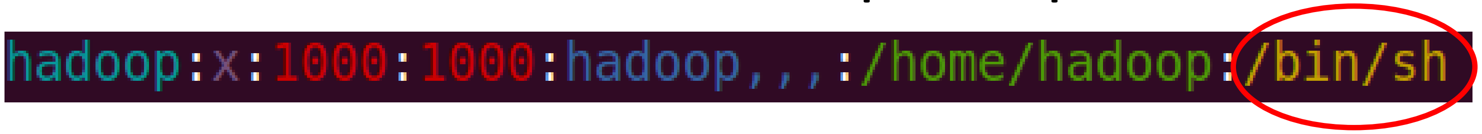
```
echo "++ Configure static IP 140.113.114.103"
ifconfig eth0 down
ifconfig eth0 140.113.114.103 netmask 255.255.255.0 up
ifconfig lo 127.0.0.1
route add default gw 140.113.114.254

hostname zedboard
```

User related files

- Files related to user setting: etc/passwd and etc/group

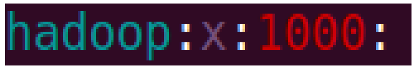
This should be "sh" or public key authentication will fail.

sudo vim ~/win/ramdisk/tmp/etc/passwd



```
hadoop:x:1000:1000:hadoop,,,:/home/hadoop:/bin/sh
```

sudo vim ~/win/ramdisk/tmp/etc/group



```
hadoop:x:1000:
```

(Note: there's no password for hadoop. However, we can ssh as hadoop using public key)

- Make the home directory for hadoop

sudo mkdir /home/hadoop

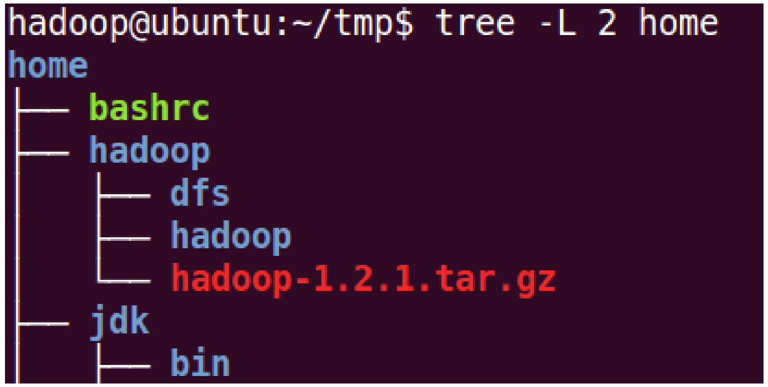
hostname resolving

```
sudo vim ~/win/ramdisk/tmp/etc/hosts
127.0.0.1 localhost
127.0.1.1 zedboard
```

```
sudo vim ~/win/ramdisk/tmp/etc/nsswitch.conf
hosts:files dns
```

Standalone Hadoop on ZedBoard

- Why standalone mode:
  - It’s kind of complicated of network setting when one machine is operating on VirtualBox.
  - We will talk about hadoop on ZedBoard using Linux machine under the same area network.
- Download “CLEAN” hadoop
  - Where “CLEAN” means no configuration is applied. This will make hadoop operating under standalone mode.
  - rename the folder to hadoop.



```
hadoop@ubuntu:~/tmp$ tree -L 2 home
home
├── bashrc
├── hadoop
│   ├── dfs
│   ├── hadoop
│   └── hadoop-1.2.1.tar.gz
├── jdk
│   ├── bin
```

Hadoop configuration

- The internal storage is not enough for hadoop. Therefore we need to mount SD card under /home. We should move the .ssh file into SD card to make public key authentication available
- We will need two partitions for SD card. Partition 1 is in FAT format. Partition 2 is in ext2(or ext3, ext4 format). The reason is that FAT format cannot support file ownership and will cause some problems when running hadoop.
  – Partition 1: The SD image for booting up.
  – Partition 2: The files that will be mounted under /home such as java, hadoop etc.
- For SD card partition, please refer to the last few pages in this slide.

Standalone Hadoop

- mount partition 2 on /home

  mount /dev/mmcblk0p2 /home
- Add java soft link

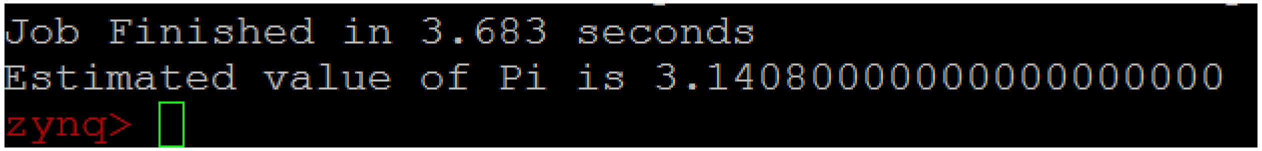
  ln –s /usr/bin/java /home/jdk/bin/java

  vi /home/hadoop/hadoop/conf/hadoop-env.sh

  export JAVA_HOME=/usr

  cd /home/hadoop/hadoop

  bin/hadoop jar hadoop-examples-1.2.1.jar pi 1 10000



```
Job Finished in 3.683 seconds
Estimated value of Pi is 3.14080000000000000000
zynq> 
```

SSH public key authentication

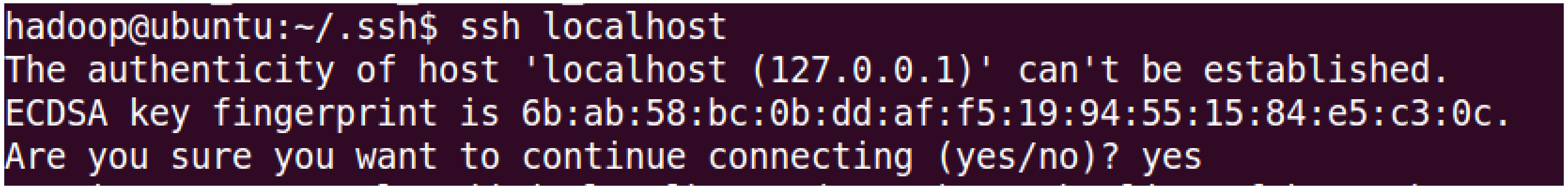
Under Linux: (work as user hadoop)

cd ~/.ssh (if not exists, mkdir one)

ssh-keygen –t rsa –P “” –f ~/.ssh/id_rsa

cat id_rsa.pub >> authorized_keys

ssh localhost



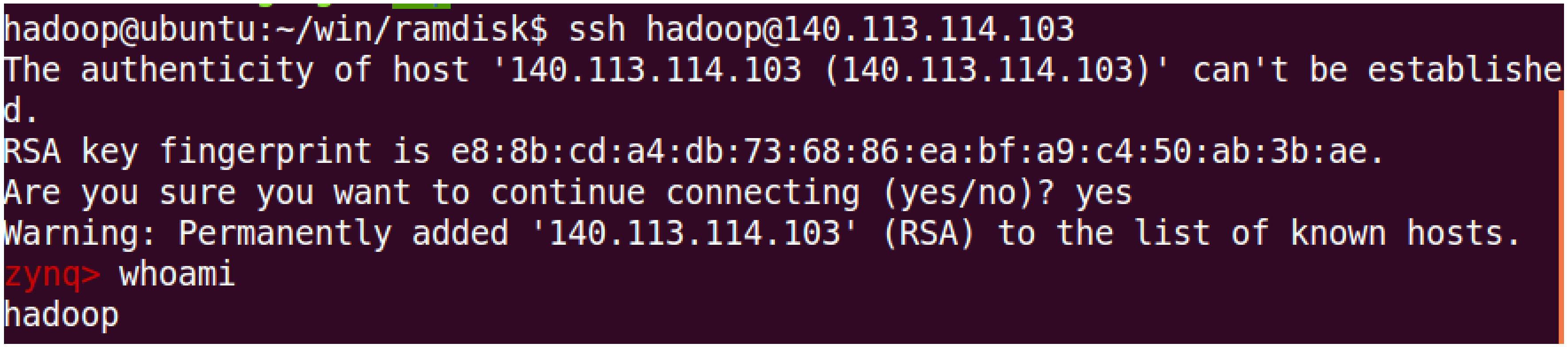
```
hadoop@ubuntu:~/.ssh$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is 6b:ab:58:bc:0b:dd:af:f5:19:94:55:15:84:e5:c3:0c.
Are you sure you want to continue connecting (yes/no)? yes
```

You will be asked the first time you login. Type yes, and if you can login without entering password, the public key authentication succeed.

mount the ramdisk on ~/win/ramdisk/tmp and then copy the authorized_keys to ZedBoard root file system.

```
mkdir ~/win/ramdisk/tmp/home/hadoop/.ssh
sudo cp ~/.ssh/authorized_keys ~/.ssh/id-rsa
~/win/ramdisk/tmp/home/hadoop/.ssh
```

Boot up and try to ssh it!



```
hadoop@ubuntu:~/win/ramdisk$ ssh hadoop@140.113.114.103
The authenticity of host '140.113.114.103 (140.113.114.103)' can't be establishe
d.
RSA key fingerprint is e8:8b:cd:a4:db:73:68:86:ea:bf:a9:c4:50:ab:3b:ae.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '140.113.114.103' (RSA) to the list of known hosts.
zynq> whoami
hadoop
```

Prepare SD card

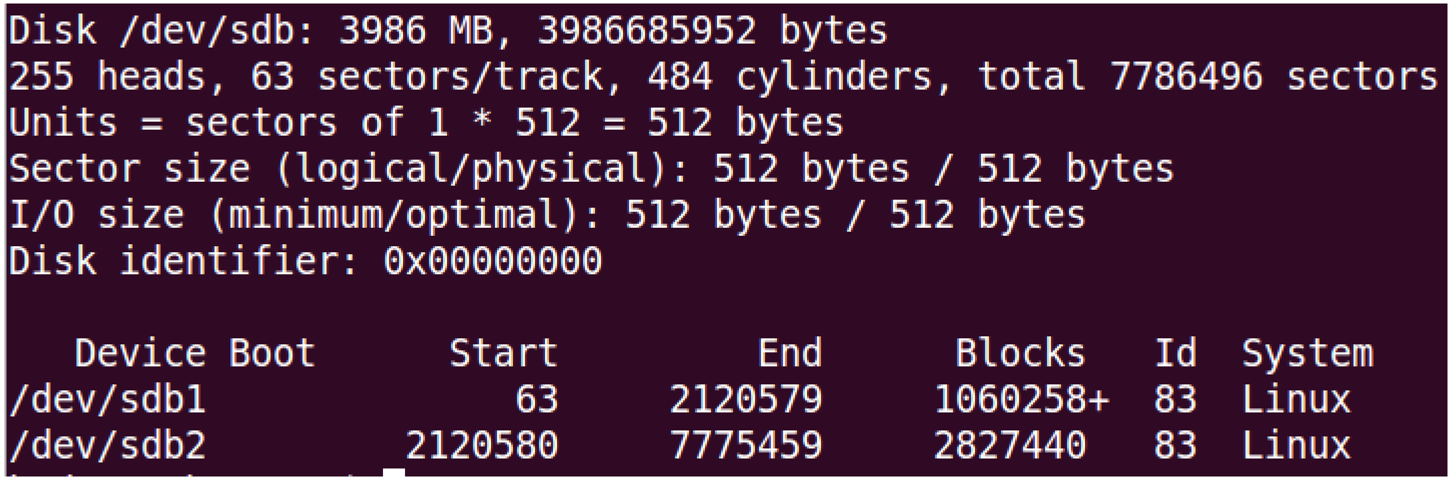
To boot up through SD cards, they should be formatted correctly.

1. The booting partition should be in FAT format.
2. For some issues, (speed, configuration … etc), we want the ext2 (ext3 or ext4) format of the remain parts.

Partition SD card by fdisk

Insert the SD card

sudo fdisk -l

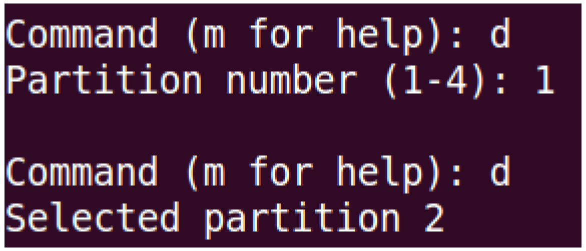


```
Disk /dev/sdb: 3986 MB, 3986685952 bytes
255 heads, 63 sectors/track, 484 cylinders, total 7786496 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1              63     2120579     1060258+  83  Linux
/dev/sdb2         2120580     7775459     2827440   83  Linux
```

This information tells us: the SD card is recognized as sdb, 3986MB in total. And there are two partitions called sdb1 and sdb2 you should umount the /dev/sdb first

sudo fdisk –c=dos –u=cylinders /dev/sdb

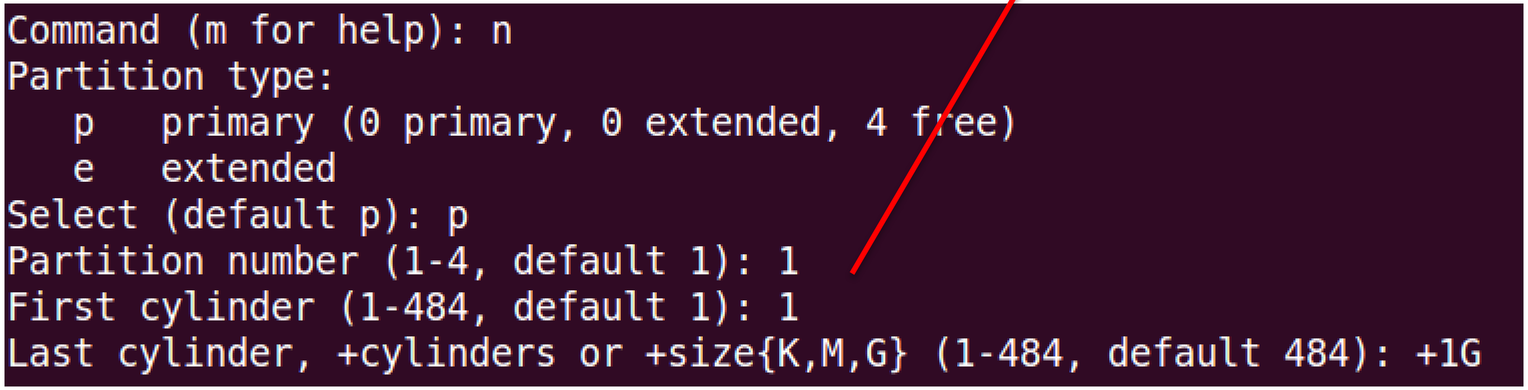
delete the original partition first



```
Command (m for help): d
Partition number (1-4): 1

Command (m for help): d
Selected partition 2
```

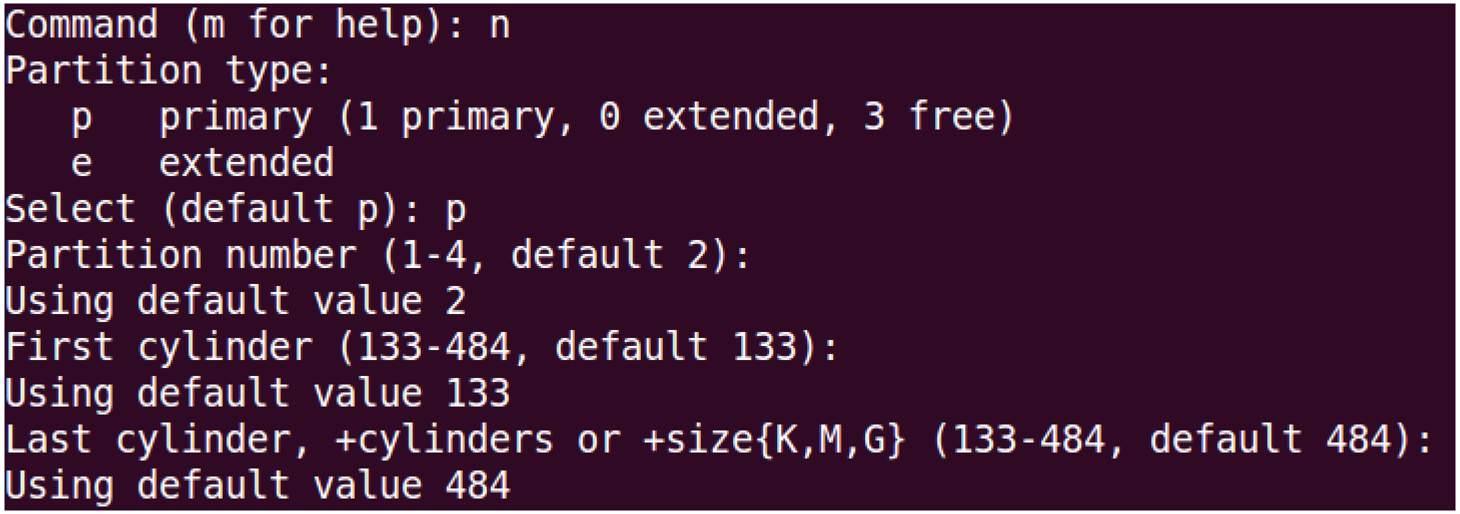
At this step press enter to use default value

new partition 1 with 1GB



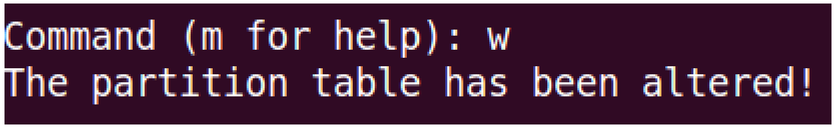
```
Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First cylinder (1-484, default 1): 1
Last cylinder, +cylinders or +size{K,M,G} (1-484, default 484): +1G
```

add the second partitions with default value.



```
Command (m for help): n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): p
Partition number (1-4, default 2):
Using default value 2
First cylinder (133-484, default 133):
Using default value 133
Last cylinder, +cylinders or +size{K,M,G} (133-484, default 484):
Using default value 484
```

write the change



```
Command (m for help): w
The partition table has been altered!
```

format these two partitions

remember: we need FAT format for partition 1

and ext4 format for partition 2

```
sudo mkfs -t vfat -n BOOT /dev/sdb1
sudo mkfs -t ext4 -L ROOT /dev/sdb2
```

you can check the new partition by fdisk