How to run diskless FreeBSD 13.2 on OrangePI Zero 2.

Whole process from scratch.
Everything is built from sources.

This guide is covering everything till the moment when we will be able ssh to root to FreeBSD on OrangePI Zero 2 (further in the document we may call it OPI).

After the installation the OrangePI Zero 2 will be able to boot from the network without any microSD card installed.

Roughly, how that particular diskless work.

It is not using PXE protocol.
It is using default boot scripts embeeded in the latest u-boot version.

The Orange PI Zero 2 hardware loads u-boot application from 2 megabytes SPI-Flash memory, which is installed onboard, and starts u-boot, then u-boot broadcasts DHCP request to the local network looking for the following information:

- Its own IP address
- IP address of TFTP server
- Filename of OS loader
- Where the root of filesystem is placed

After u-boot loads OS loader from TFTP server it starts the loader passing URI of root of filesystem to it then loader loads the FreeBSD kernel from the root URI, which is NFS server, and starts the kernel passing root URI to it then kernel initialises the system and starts it and works using root filesystem which is placed on NSF server as it would be a usual local disk.

IMPORTANT: For simplify the things this installation considers that the root filesystem on the dedicated directory on the NFS server will be used exclusevely only by one Orange PI Zero 2. For shared diskless installation we will need slightly more complicated instructions, probably, it will be posted later.

What we will need.

We will need the following:

- PC architecture amd64, 2 core processor, 4GB RAM, 50GB HDD with microSD card adapter. Just note: that PC will keep all the files for FreeBSD OrangePI Zero 2 diskless system.
- OrangePI Zero 2
- microSD card 100MB or more, ideally it should be SanDisk, like that. **Other cards can work but unstable, not recomended!**
- Local ethernet IP network with connection to internet.
- Our own DHCP server on the network with access to configure it.
- Two dedicated static IP in our network: for PC and for OPI and two hostnames for them, unique in our network.

IMPORTANT NOTE: everywhere in the document we must replace ${{SOME_TEXT}} to the thing which is described by SOME_TEXT, example: ${{OPI_IP}} must be replaced to IP address of OPI.

1. Prepare FreeBSD build machine on PC.

1.1. Get image of installatin CD from here:

FreeBSD 13.2

1.2. Install FreeBSD on the PC from the CD image.

- Answer all installation questions as default.
- Set static IP address: ${{PC_IP}}
- Additional services started at boot: ntpdate, ntpd.
- Create additional username: "build" invite to additional group "wheel"

1.3 Continue on freshly installed FreeBSD 13.2:

ssh to build@${{PC_IP}} and enter the commands, answering "Y" on all questions:

```
$ su -
# freebsd-update fetch
# freebsd-update install
# reboot
```

After PC reboot, again, ssh to build@${{PC_IP}}, enter the following, answering "Y" on all questions.

```
$ su -
# pkg upgrade
# pkg install bash sudo git gmake bison gcc aarch64-none-elf-gcc python3 py39-pip swig
```

In the next command: find the string "# %wheel ALL=(ALL:ALL) ALL" and remove "# "

```
# visudo
```

Enter the following commands:

```
# bash
# echo "${{PC_IP}} $(hostname)" >> /etc/hosts
```

Great! Everything is ready now to build the FreeBSD from sources for OrangePI Zero 2 which has architecture arm64.

2. Obtain sources of FreeBSD 13.2, make patch for OrangePI Zero 2 and build everything.

2.1. Obtain the sources

ssh to build@${{PC_IP}} and execute the commands:

```
$ bash
$ cd
$ mkdir freebsd_13_2
$ cd freebsd_13_2
$ export BASEDIR=$(pwd)
```

```
$ export MAKEOBJDIRPREFIX=$BASEDIR/obj

$ git clone https://github.com/freebsd/freebsd.git src.main

$ cd src.main

$ git checkout 51117ed1 # Just to be sure that we are patching the same version as it was at the moment of creating the document.

$ cd ..

$ git clone https://github.com/freebsd/freebsd.git src

$ cd src

$ git checkout release/13.2.0
```

2.2. Make the patch for build dtb-file which will see the ethernet interface of OrangePI Zero 2

Note: FreeBSD 13.2 does not oficially support Orange PI Zero 2, so, we have to do that patch.
The patch will allow FreeBSD 13.2 to see only ethernet interface, USB and microSD card will still be unavailable.
We are making diskless station, so, only ethernet will be enough for us.

Continue entering commands on the same command line as before:

```
$ SD=$BASEDIR/src.main/sys/contrib/device-tree

$ cd sys/contrib/device-tree/src/arm64/allwinner

$ cp $SD/src/arm64/allwinner/sun50i-h616-orangepi-zero2.dts .

$ echo '12a13,15
> #define RST_BUS_EMAC 33
> #define CLK_BUS_EMAC 84
>
128c131
<                         compatible = "allwinner,sun50i-h616-ccu";
---
>                         compatible = "allwinner,sun50i-h6-ccu";
493c496
<                         clocks = <&ccu CLK_BUS_EMAC0>;
---
>                         clocks = <&ccu CLK_BUS_EMAC>;
495c498
<                         resets = <&ccu RST_BUS_EMAC0>;
---
>                         resets = <&ccu RST_BUS_EMAC>;
508c511
<                         compatible = "allwinner,sun50i-h616-rtc";
---
>                         compatible = "allwinner,sun50i-h6-rtc";
510a514
>                         clock-output-names = "osc32k", "osc32k-out", "iosc";' | patch $SD/src/arm64/allwinner/sun50i-h616.dtsi -o sun50i-h616.dtsi

$ cd ../../../include/dt-bindings/clock

$ cp $SD/include/dt-bindings/clock/sun50i-h616-ccu.h .

$ cp $SD/include/dt-bindings/clock/sun6i-rtc.h .

$ cp $SD/include/dt-bindings/clock/sun50i-h6-r-ccu.h .

$ cd ../reset

$ cp $SD/include/dt-bindings/reset/sun50i-h616-ccu.h .

$ cp $SD/include/dt-bindings/reset/sun50i-h6-r-ccu.h .

$ cd $BASEDIR/src/sys/modules/dtb/allwinner

$ echo '56a57
>       allwinner/sun50i-h616-orangepi-zero2.dts \' | patch Makefile -o Makefile.patched

$ mv Makefile.patched Makefile
```

2.3. Build everything and put the root of the system into /exports/orangepizero2-${{HOSTNAME_OF_OPI}}

Continue entering commands on the same command line as before:

```
$ DESTDIR=/exports/orangepizero2-${{HOSTNAME_OF_OPI}}

$ NJOBS=$(sysctl hw.ncpu|awk '{print $2-1}')

$ cd $BASEDIR/src

$ make -j$NJOBS buildworld TARGET_ARCH=aarch64

$ make -j$NJOBS buildkernel TARGET_ARCH=aarch64 KERNCONF=GENERIC

$ sudo mkdir -p $DESTDIR

$ sudo -E make installworld TARGET_ARCH=aarch64 DESTDIR=$DESTDIR

$ sudo -E make distribution TARGET_ARCH=aarch64 DESTDIR=$DESTDIR

$ sudo -E make installkernel TARGET_ARCH=aarch64 KERNCONF=GENERIC DESTDIR=$DESTDIR
```

3. Additional settings for diskless clients.

These settings will be done in the diskless clients root filesystem which we have placed at /exports/orangepizero2-${{HOSTNAME_OF_OPI}}

3.1. Creating general configuration for for diskless clients.

Continue entering commands on the same command line as before:

```
$ sudo bash

# echo '${{OPI_IP}} ${{HOSTNAME_OF_OPI}}' >> /etc/hosts

# cp /etc/resolv.conf /exports/orangepizero2-${{HOSTNAME_OF_OPI}}/etc
```

```
# echo 'sshd_enable="YES" # We need ssh access
background_fsck="NO"      # NEVER run fsck on nfs mounted partitions
nfs_client_enable="YES"   # The diskless client is an NFS client
rpc_lockd_enable="YES"    # Some application require file locking
rpc_statd_enable="YES"    # Some application require file locking
ntpdate_enable="YES"      # We need actual date/time
ntpd_enable="YES"' > /exports/orangepizero2-${{HOSTNAME_OF_OPI}}/etc/rc.conf

# echo '# Device                                          Mount  FStype  Options       Dump  Pass
${{PC_IP}}:/exports/orangepizero2-${{HOSTNAME_OF_OPI}} /      nfs     rw            0     0
proc                                                  /proc  procfs  rw            0     0' > /exports/orangepizero2-${{HOSTNAME_OF_OPI}}/etc/fstab
```

3.2. Allow root to do remote login to Orange PI Zero 2.

After everything will be started, we need remote root access to our OPI for continue the configuration.
So, we are changing sshd configuration and putting our ssh public key to the OPI root home directory.
Continue entering commands on the same command line as before:

```
# cd /exports/orangepizero2-${{HOSTNAME_OF_OPI}}/etc/ssh

# cat sshd_config | sed 's/#PermitRootLogin no/PermitRootLogin yes/g' > sshd_config.new

# mv sshd_config.new sshd_config

# cd /exports/orangepizero2-${{HOSTNAME_OF_OPI}}/root

# mkdir .ssh

# cd .ssh

# echo '${{OUR_SSH_PUBLIC_KEY}}' > authorized_keys

# exit
```

3.3. Script for u-boot to let us know that the boot process has been started.

Continue entering commands on the same command line as before:

```
$ cd ~/u-boot

$ echo 'gpio set 76;gpio clear 77' > boot.scr

$ tools/mkimage -A arm -T script -d boot.scr boot.scr.uimg

$ sudo cp boot.scr.uimg /exports/orangepizero2-${{HOSTNAME_OF_OPI}}/boot

$ exit

$ exit
```

4. Preparing Orange PI Zero 2 for diskless boot.

4.1. Create u-boot binary file and script for writing to SPI-Flash.

ssh to build@${{PC_IP}}

4.1.1 Create firmware binary file bl31.

Execute commands:

```
$ bash

$ cd

$ mkdir bl31

$ cd bl31

$ git clone https://github.com/ARM-software/arm-trusted-firmware.git

$ cd arm-trusted-firmware

$ git checkout v2.8.0

$ gmake realclean

$ TF_LDFLAGS="--no-warn-rwx-segment" gmake CROSS_COMPILE=aarch64-none-elf- PLAT=sun50i_h616 bl31
```

4.1.2. Obtain u-boot sources and build u-boot binary.

Continue entering commands on the same command line as before:

```
$ cd

$ mkdir u-boot

$ git clone https://source.denx.de/u-boot/u-boot.git

$ cd u-boot

$ git checkout v2023.04

$ gmake orangepi_zero2_defconfig

$ cp ~/bl31/arm-trusted-firmware/build/sun50i_h616/release/bl31.bin .

$ gmake CROSS_COMPILE=aarch64-none-elf-
```

4.1.3. Prepare the script for write u-boot binary to OPI SPI-Flash memory.

Continue entering commands on the same command line as before:

```
$ echo 'bootcmd=gpio set 76;sf probe;sf erase 0 0x200000;load mmc 0 0x42000000 u-boot-sunxi-with-spl.bin;sf write 0x42000000 0 0x200000;while true;do gpio toggle 7

$ tools/mkenvimage -s 65536 -o uboot.env uboot.txt

$ exit

$ exit
```

4.2. Prepare microSD card.

This card will be used only once but for each OPI which we would like to run on our network.
So, create it, use it and keep it for future when we need to initialise others OPI for work as diskless station.
We have to create a FAT partition on the card where we will put u-boot image file and the writing script, also we have to put u-boot image to special non-partitioned space on the card where OPI hardware can find it and start to boot.
That is VERY importand that card MUST be empty, without any partitions on it.
So, to make sure that it is empty we cleanup it first.

Insert the card into PC and do ssh to build@${{PC_IP}} and enter the following commands:

```
$ cd ~/u-boot

$ sudo bash

# SD=${{path_to_microSD_card_device}}

# gpart destroy -F $SD #Here may be error if card already empty

# dd if=/dev/zero of=$SD bs=1024 count=4096
```

So, card is empty now and we can create everything what we need.
Continue entering commands on the same command line as before:

```
# gpart create -s mbr $SD

# gpart add -a 3M -t fat16 -s 20M $SD #Started from 3mbyte, leaving space for u-boot

# newfs_msdos -F 16 ${SD}s1

# dd if=u-boot-sunxi-with-spl.bin of=$SD bs=1k seek=8 conv=sync

# mount_msdosfs ${SD}s1 /mnt

# cp u-boot-sunxi-with-spl.bin /mnt

# cp uboot.env /mnt

# umount /mnt

# exit

$ exit
```

4.3. Write u-boot image to OPI SPI-Flash memory.

Great! Initialisation card is ready and we can initialise our very first Orange PI Zero 2.

That is very simple now, just insert the card into OPI and turn it on.

In about 10 seconds after turning OPI on we should see on OPI the red led ligting it does mean that the flash process has been started!

In about a minute after turning OPI on we should see on OPI the red and green leds are blinking, it does mean that the flash process has been finished, so, we can turn OPI off and eject the microSD card.

4.4. Configure OPI to boot from SPI-Flash.

VERY IMPORTANT! We have to inform our OPI that it can boot now from SPI-flash.
To do that we have to install a jumper between two pin contacts: 13 and 14.
They are in the very middle of 26-pin sets.
With the jumper between 13 and 14 the OPI will try to boot from microSD card first and if microSD card does not exists it will try to boot from SPI flash.

Ok! Congratulations! Our OPI is ready to start diskless booting!
Is that all?
Not quite...

4.5. Discovering of MAC-address of OPI.

Now, we need to discover what the MAC-address our OPI has.
For do that we have to ssh to build@${{PC_IP}} and enter the following command:

```
$ sudo tcpdump -i em0 port 67
```

It will stuck here and will display all DHCP requests which will appear in our network.

After run that command, connect the OPI (without the microSD card) to the network and turn it on.

Wait about 30 seconds and we should see on the terminal (where we started the tcpdump) something like that:
"... IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP, Request from 02:00:xx:xx:xx:xx (oui Unknown), length 300"

So, the MAC address of our OPI is 6 hex numbers after **"Request from"**.

Highly likely it will start from 02:00 but may be something else, who knows...

Ok. We have discovered the MAC address of our OPI. Write it down, we will need it in future and turn the OPI off.

5. Preparing the network infrastructure.

Be patient! We are almost there!

5.1. Setup TFTP and NFS servers on PC:

ssh to build@${{PC_IP}} and execute the commands:

```
$ sudo bash

# echo 'inetd_enable="YES"' >> /etc/rc.conf

# echo 'tftp dgram udp wait root /usr/libexec/tftpd tftpd -l -s /exports/orangepizero2-${{HOSTNAME_OF_OPI}}/boot' >> /etc/inetd.conf
```

```
# service inetd start

# echo 'rpcbind_enable="YES"
nfs_server_enable="YES"
mountd_enable="YES"
rpc_lockd_enable="YES"
rpc_statd_enable="YES"' >> /etc/rc.conf

# echo '/exports/orangepizero2-${{HOSTNAME_OF_OPI}} -maproot=root:wheel ${{OPI_IP}} >> /etc/exports

# service nfsd start

# service lockd start

# exit

$ exit
```

5.2. Configure DHCP server:

Go to our DHCP server and add configuration required for network boot of our Orange PI Zero 2.
That configuration can vary depends on type DHCP server is used.
Here is an example of configuration of DHCP server working under Linux CentOS 6.
Add the following lines to /etc/dhcp/dhcpd.conf:

```
host ${{HOSTNAME_OF_OPI}} {
    hardware ethernet ${{MAC_ADDRESS_WHICH_WAS_DISCOVERED_IN_P4.5}};
    fixed-address ${{OPI_IP}};
    next-server ${{PC_IP}}; # tftp server
    filename "loader.efi";
    option host-name "${{HOSTNAME_OF_OPI}}";
    option root-path "${{PC_IP}}:/exports/orangepizero2-${{HOSTNAME_OF_OPI}}"; #Where the root of filesystem is placed
}
```

Restart the dhcp server. On the CentOS 6 it can be done by command under root:
service dhcpd restart

6. Testing.

Is that all now???
Yes! Eventually we are here!
Connect our Orange PI Zero 2 withour microSD card to the network and turn it on.
If everything is right then in about 10 seconds we can see red light on Orange PI Zero 2, taht is mean the boot process has been started.
In about a minute after the red light appear you should be able to ssh as root to diskless Orange PI Zero 2!

Horray!! We did it!!!

Postscriptum:
If the red light is not appearing in one minute, try to turn it off and on again.
Try it two-three of times.
If that still does not work, try another Orange PI Zero 2.
I got the experience that some of my OPIs does not catch DHCP configuration. I do not know why.
If that still does not work, connect the serial console to the Orange PI Zero 2 and see what is going on.