Lab 4: Booting the Linux Kernel

Abhinav Subramanian

ECEN 449-502

Due date: 2/18/2020

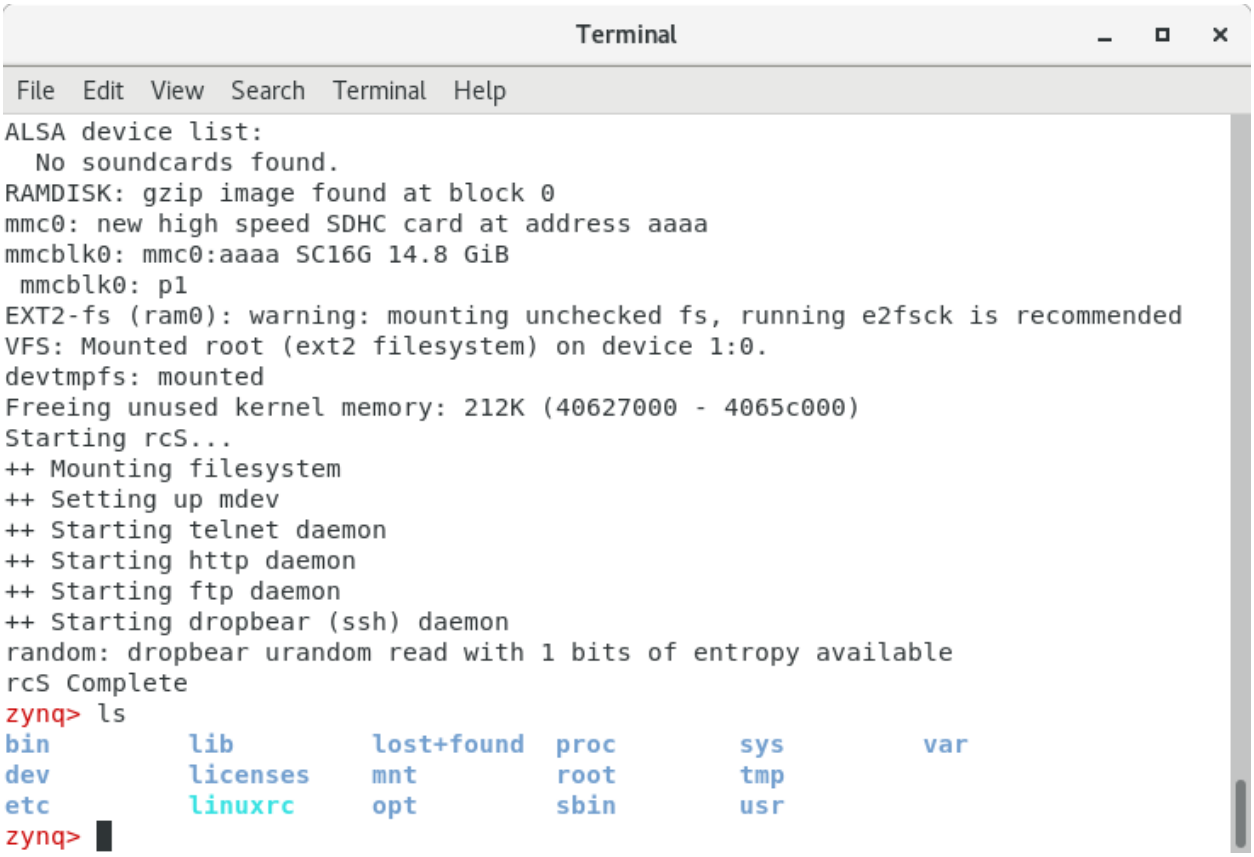
Introduction:

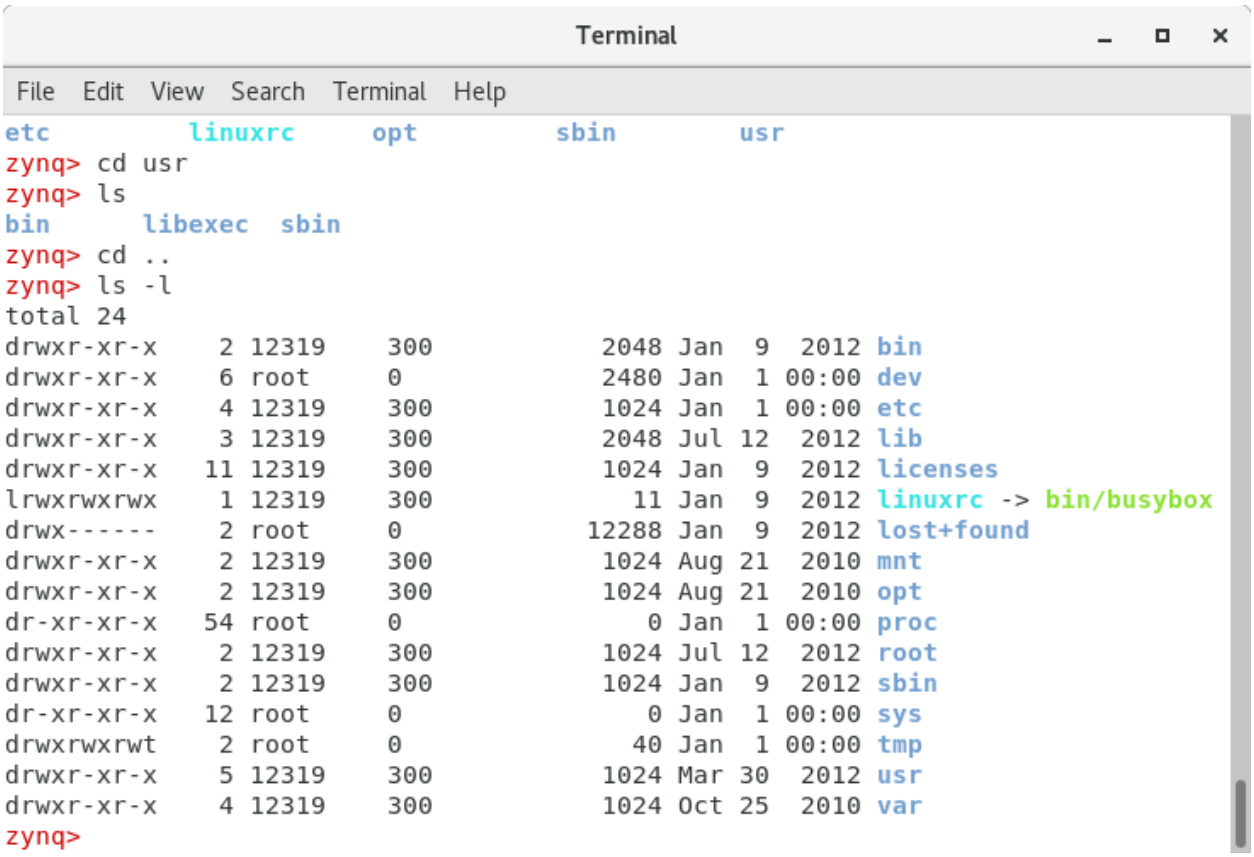
The aim of this lab was to learn how to boot the Linux OS on the Zybo board. We used Vivado to build the microprocessor system, then then we created a boot image for the system. In addition to this, we had to create a temporary file system called the ramdisk that could be used for booting, and finally we had to create the device tree to provide information about the peripherals in the system.

Procedure:

- Create a new block diagram in Vivado and create the microprocessor design with the multiply IP created in lab 3. This will allow us to run Linux on the ZYNQ7 processing system. Once finished, generate the bitstream.
- Untar the U-boot file and configure and build the files for the ZYBO. Make sure to source settings64.sh.
- Go into the SDK to combine the bitstream with the bootloader and u-boot executable file. This will generate the BOOT.bin file that you will use to boot Linux.
- Untar the Linux kernel, configure and compile it. Then, we have to convert the kernel image from a zipped to an unzipped image (uImage) to be able to boot Linux on the zybo.
- Edit the devicetree.dts file to include the Multiply IP, then we need to convert it to a .dtb file to boot.
- Wrap the ramdisk image to create uramdisk.image.gz
- Copy BOOT.bin, uImage, uramdisk.image.gz and devicetree.dtb onto the SD card, and boot linux using picocom.

Results:


Terminal
File Edit View Search Terminal Help
ALSA device list:
No soundcards found.
RAMDISK: gzip image found at block 0
mmc0: new high speed SDHC card at address aaaa
mmcblk0: mmc0:aaaa SC16G 14.8 GiB
mmcblk0: p1
EXT2-fs (ram0): warning: mounting unchecked fs, running e2fsck is recommended
VFS: Mounted root (ext2 filesystem) on device 1:0.
devtmpfs: mounted
Freeing unused kernel memory: 212K (40627000 - 4065c000)
Starting rcS...
++ Mounting filesystem
++ Setting up mdev
++ Starting telnet daemon
++ Starting http daemon
++ Starting ftp daemon
++ Starting dropbear (ssh) daemon
random: dropbear urandom read with 1 bits of entropy available
rcS Complete
zynq> ls
bin lib lost+found proc sys var
dev licenses mnt root tmp
etc linuxrc opt sbin usr
zynq>



Conclusion:

In this lab, I learned how to boot the Linux OS on the Zybo board. After building and compiling all the necessary files, I booted Linux and was able to navigate through all the file directories. I didn't really run into any setbacks, but the files did take forever to compile, especially the Linux kernel.

Postlab questions:

1. Yes, this local memory is called the cache. It exists in close proximity, if not directly in, the CPU.
2. I can write to everything except sys and proc. After a reboot, the created files are lost, as the files are stored on the ramdisk (the temporary file system), which is volatile.
3. If we added a new peripheral, we'd have to regenerate the bitstream, and this would hence change BOOT.bin. The uImage and ramdisk would remain unchanged. However, the .dts file would need a new entry describing the new peripheral.

Appendix:

No additional code was written for this lab.