Student ID: 406410035
Name: 秦紫頤
E-mail: chinjoyce30@gmail.com

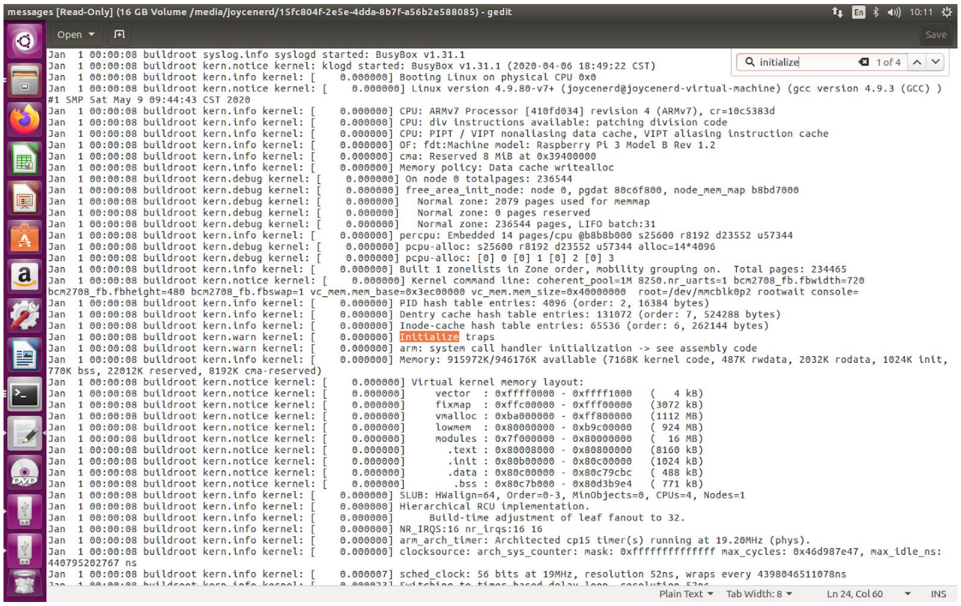
Lab Title: Exception

Lab Purpose:

First, observe the initialization of the exception. Second, observe the implementation of system calls. Third, add a new system call.

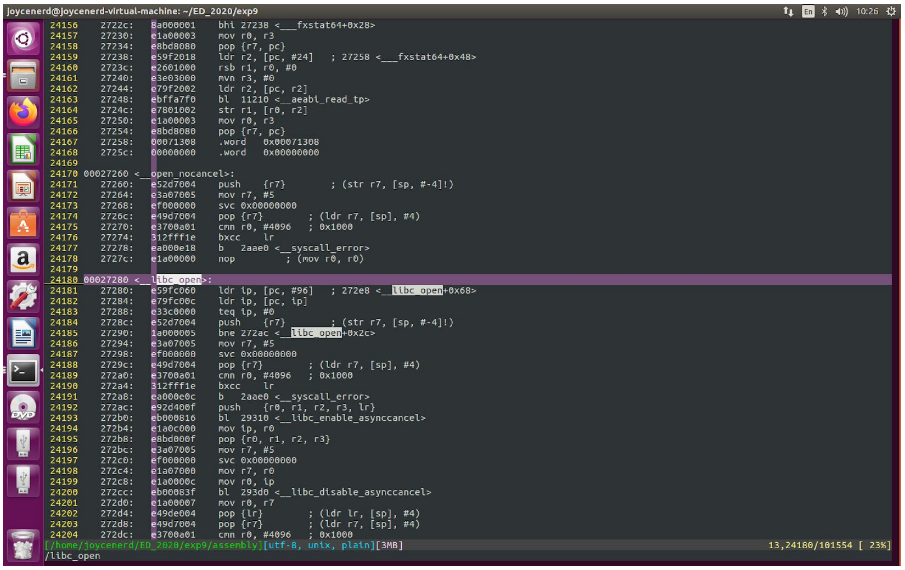
Lab Procedure:

1. Observe exception handle
    a. Download Linux kernel source code for Raspberry Pi
    b. Add a printk statement **"Initialize traps\n"** into ~/pi_kernel/linux/init/main.c
    c. Add a printk statement **"arm: system call handler initialization -> see assembly code\n"** in the function void __init trap_init(void)
    d. Compile the kernel:
        i. cd ~/pi_kernel/linux
        ii. make ARCH=arm bcm2709_defconfig
        iii. make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bzImage
    e. Copy the compiled zImage to SD card
        i. Plug the SD card into your laptop
        ii. mount /dev/sdb1 ~/mmc1
        iii. sudo rm ~/mmc1/zImage
        iv. cd ~/pi/kernel/linux/arch/arm/boot
        v. sudo cp -rf zImage ~/mmc1
        vi. sudo umount ~/mmc1
    f. Put the SD card back to Pi and power on
    g. dmesg


messages [Read-Only] (16 GB Volume /media/joycenerd/15fc804f-2e5e-4dda-8b7f-a56b2e588085) - gedit
En
10:11
Open
Save
initialize
1 of 4
Plain Text
Tab Width: 8
Ln 24, Col 60
INS

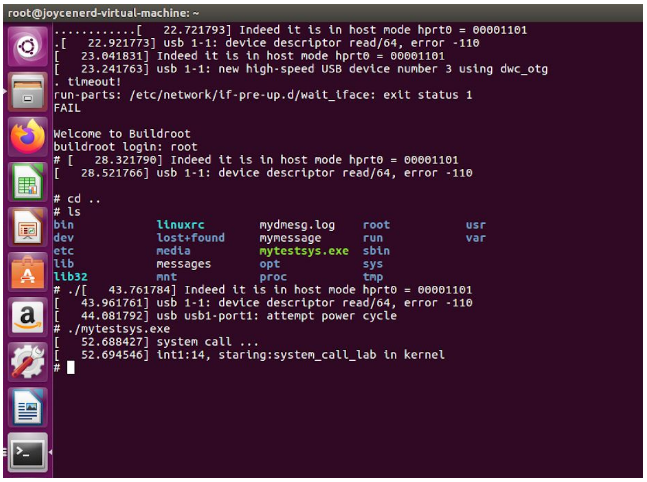
```
Jan  1 00:00:08 buildroot syslog.info syslogd started: BusyBox v1.31.1
Jan  1 00:00:08 buildroot kern.notice kernel: klogd started: BusyBox v1.31.1 (2020-04-06 18:49:22 CST)
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000000] Booting Linux on physical CPU 0x0
Jan  1 00:00:08 buildroot kern.notice kernel: [    0.000000] Linux version 4.9.80-v7+ (joycenerd@joycenerd-virtual-machine) (gcc version 4.9.3 (GCC) )
#1 SMP Sat May 9 09:44:43 CST 2020
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000000] CPU: ARMv7 Processor [410fd034] revision 4 (ARMv7), cr=10c5383d
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000000] CPU: div instructions available: patching division code
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000000] CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000000] OF: fdt:Machine model: Raspberry Pi 3 Model B Rev 1.2
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000000] cma: Reserved 8 MiB at 0x39400000
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000000] Memory policy: Data cache writealloc
Jan  1 00:00:08 buildroot kern.debug kernel: [    0.000000] On node 0 totalpages: 236544
Jan  1 00:00:08 buildroot kern.debug kernel: [    0.000000] free_area_init_node: node 0, pgdat 80c6f800, node_mem_map b8bd7000
Jan  1 00:00:08 buildroot kern.debug kernel: [    0.000000]   Normal zone: 2079 pages used for memmap
Jan  1 00:00:08 buildroot kern.debug kernel: [    0.000000]   Normal zone: 0 pages reserved
Jan  1 00:00:08 buildroot kern.debug kernel: [    0.000000]   Normal zone: 236544 pages, LIFO batch:31
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000000] percpu: Embedded 14 pages/cpu @b8b8b000 s25600 r8192 d23552 u57344
Jan  1 00:00:08 buildroot kern.debug kernel: [    0.000000] pcpu-alloc: s25600 r8192 d23552 u57344 alloc=14*4096
Jan  1 00:00:08 buildroot kern.debug kernel: [    0.000000] pcpu-alloc: [0] 0 [0] 1 [0] 2 [0] 3
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000000] Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 234465
Jan  1 00:00:08 buildroot kern.notice kernel: [    0.000000] Kernel command line: coherent_pool=1M 8250.nr_uarts=1 bcm2708_fb.fbwidth=720
bcm2708_fb.fbheight=480 bcm2708_fb.fbswap=1 vc_mem.mem_base=0x3ec00000 vc_mem.mem_size=0x40000000  root=/dev/mmcblk0p2 rootwait console=
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000000] PID hash table entries: 4096 (order: 2, 16384 bytes)
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000000] Dentry cache hash table entries: 131072 (order: 7, 524288 bytes)
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000000] Inode-cache hash table entries: 65536 (order: 6, 262144 bytes)
Jan  1 00:00:08 buildroot kern.warn kernel: [    0.000000] Initialize traps
Jan  1 00:00:08 buildroot kern.warn kernel: [    0.000000] arm: system call handler initialization -> see assembly code
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000000] Memory: 915972K/946176K available (7168K kernel code, 487K rwdata, 2032K rodata, 1024K init,
770K bss, 22012K reserved, 8192K cma-reserved)
Jan  1 00:00:08 buildroot kern.notice kernel: [    0.000000] Virtual kernel memory layout:
Jan  1 00:00:08 buildroot kern.notice kernel: [    0.000000]     vector  : 0xffff0000 - 0xffff1000   (   4 kB)
Jan  1 00:00:08 buildroot kern.notice kernel: [    0.000000]     fixmap  : 0xffc00000 - 0xfff00000   (3072 kB)
Jan  1 00:00:08 buildroot kern.notice kernel: [    0.000000]     vmalloc : 0xba000000 - 0xff800000   (1112 MB)
Jan  1 00:00:08 buildroot kern.notice kernel: [    0.000000]     lowmem  : 0x80000000 - 0xb9c00000   ( 924 MB)
Jan  1 00:00:08 buildroot kern.notice kernel: [    0.000000]     modules : 0x7f000000 - 0x80000000   (  16 MB)
Jan  1 00:00:08 buildroot kern.notice kernel: [    0.000000]       .text : 0x80008000 - 0x80800000   (8160 kB)
Jan  1 00:00:08 buildroot kern.notice kernel: [    0.000000]       .init : 0x80b00000 - 0x80c00000   (1024 kB)
Jan  1 00:00:08 buildroot kern.notice kernel: [    0.000000]       .data : 0x80c00000 - 0x80c79cbc   ( 488 kB)
Jan  1 00:00:08 buildroot kern.notice kernel: [    0.000000]        .bss : 0x80c7b000 - 0x80d3b9e4   ( 771 kB)
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000000] SLUB: HWalign=64, Order=0-3, MinObjects=0, CPUs=4, Nodes=1
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000000] Hierarchical RCU implementation.
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000000]      Build-time adjustment of leaf fanout to 32.
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000000] NR_IRQS:16 nr_irqs:16 16
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000000] arm_arch_timer: Architected cp15 timer(s) running at 19.20MHz (phys).
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000000] clocksource: arch_sys_counter: mask: 0xffffffffffffff max_cycles: 0x46d987e47, max_idle_ns:
440795202767 ns
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000007] sched_clock: 56 bits at 19MHz, resolution 52ns, wraps every 4398046511078ns
Jan  1 00:00:08 buildroot kern.info kernel: [    0.000023] Switching to timer-based delay loop, resolution 52ns
```

2. Observe the system call mechanism
    a. Write a simple hello world program using C
    b. Use cross compiler to compile your program: arm-linux-gnueabihf-gcc -static hello.c -o hello.exe
    c. Disassemble the program: arm-linux-gnueabihf-objdump -d hello.exe > assembly



```
   2722c:	8a000001 	bhi	27238 <___fxstat64+0x28>
   27230:	e1a00003 	mov	r0, r3
   27234:	e8bd8080 	pop	{r7, pc}
   27238:	e59f2018 	ldr	r2, [pc, #24]	; 27258 <___fxstat64+0x48>
   2723c:	e2601000 	rsb	r1, r0, #0
   27240:	e3e03000 	mvn	r3, #0
   27244:	e79f2002 	ldr	r2, [pc, r2]
   27248:	ebffa7f0 	bl	11210 <__aeabi_read_tp>
   2724c:	e7801002 	str	r1, [r0, r2]
   27250:	e1a00003 	mov	r0, r3
   27254:	e8bd8080 	pop	{r7, pc}
   27258:	00071308 	.word	0x00071308
   2725c:	00000000 	.word	0x00000000

00027260 <__open_nocancel>:
   27260:	e52d7004 	push	{r7}		; (str r7, [sp, #-4]!)
   27264:	e3a07005 	mov	r7, #5
   27268:	ef000000 	svc	0x00000000
   2726c:	e49d7004 	pop	{r7}		; (ldr r7, [sp], #4)
   27270:	e3700a01 	cmn	r0, #4096	; 0x1000
   27274:	312fff1e 	bxcc	lr
   27278:	ea000e18 	b	2aae0 <__syscall_error>
   2727c:	e1a00000 	nop			; (mov r0, r0)

00027280 <__libc_open>:
   27280:	e59fc060 	ldr	ip, [pc, #96]	; 272e8 <__libc_open+0x68>
   27284:	e79fc00c 	ldr	ip, [pc, ip]
   27288:	e33c0000 	teq	ip, #0
   2728c:	e52d7004 	push	{r7}		; (str r7, [sp, #-4]!)
   27290:	1a000005 	bne	272ac <__libc_open+0x2c>
   27294:	e3a07005 	mov	r7, #5
   27298:	ef000000 	svc	0x00000000
   2729c:	e49d7004 	pop	{r7}		; (ldr r7, [sp], #4)
   272a0:	e3700a01 	cmn	r0, #4096	; 0x1000
   272a4:	312fff1e 	bxcc	lr
   272a8:	ea000e0c 	b	2aae0 <__syscall_error>
   272ac:	e92d400f 	push	{r0, r1, r2, r3, lr}
   272b0:	eb000816 	bl	29310 <__libc_enable_asynccancel>
   272b4:	e1a0c000 	mov	ip, r0
   272b8:	e8bd000f 	pop	{r0, r1, r2, r3}
   272bc:	e3a07005 	mov	r7, #5
   272c0:	ef000000 	svc	0x00000000
   272c4:	e1a07000 	mov	r7, r0
   272c8:	e1a0000c 	mov	r0, ip
   272cc:	eb00083f 	bl	293d0 <__libc_disable_asynccancel>
   272d0:	e1a00007 	mov	r0, r7
   272d4:	e49de004 	pop	{lr}		; (ldr lr, [sp], #4)
   272d8:	e49d7004 	pop	{r7}		; (ldr r7, [sp], #4)
   272dc:	e3700a01 	cmn	r0, #4096	; 0x1000
[/home/joycenerd/ED_2020/exp9/assembly][utf-8, unix, plain][3MB]                13,24180/101554 [ 23%]
/libc_open
```

3. Write your own system call
    a. Add **"CALL(sys_mysyscall)"** to ~/pi_kernel/linux/arch/arm/kernel/calls.S
    b. Add define statement of the system call in unistd.h: **#define __NR_mysyscall (__NR_SYSCALL_BASE+397)**
    c. Write the content of the new added system call into ~/pi-kernel/linux/arch/arm/kernel/mysyscall.c
    d. Add the function declaration in syscalls.h: **asmlinkage void sys_mysyscall(int a, char* b)**;
    e. Modified the Makefile in ~/pi_kernel/linux/arch/arm/kernel: add **"my_syscall.o"** at the tail of **obj-y**
    f. Compile the kernel again and copy the zImage to the SD card
4. Test the system call
    a. Write a user program (mytestsys.c) to test call the system call I added in the previous part
    b. Use cross compiler to compile the program arm-linux-gnueabihf-gcc -I ~/pi-kernel/linux/include/ -static -g mytestsys.c -o mytestsys.exe
    c. Copy mytestsys.exe into the second partition of the SD card
    d. Put the SD card back into Pi and power on
    e. Execute mytestsys.exe to see the system call we added has worked or not



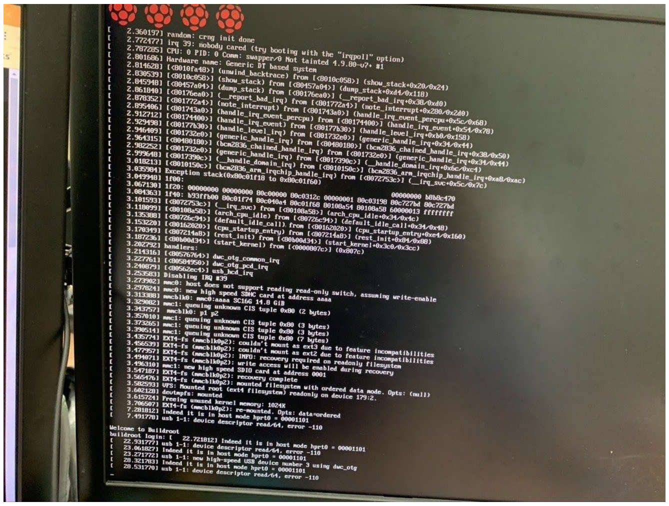
```
............[   22.721793] Indeed it is in host mode hprt0 = 00001101
.[   22.921773] usb 1-1: device descriptor read/64, error -110
[   23.041831] Indeed it is in host mode hprt0 = 00001101
[   23.241763] usb 1-1: new high-speed USB device number 3 using dwc_otg
. timeout!
run-parts: /etc/network/if-pre-up.d/wait_iface: exit status 1
FAIL

Welcome to Buildroot
buildroot login: root
# [   28.321790] Indeed it is in host mode hprt0 = 00001101
[   28.521766] usb 1-1: device descriptor read/64, error -110

# cd ..
# ls
bin           linuxrc       mydmesg.log   root          usr
dev           lost+found    mymessage     run           var
etc           media         mytestsys.exe sbin
lib           messages      opt           sys
lib32         mnt           proc          tmp
# ./[   43.761784] Indeed it is in host mode hprt0 = 00001101
[   43.961761] usb 1-1: device descriptor read/64, error -110
[   44.081792] usb usb1-port1: attempt power cycle
# ./mytestsys.exe
[   52.688427] system call ...
[   52.694546] int1:14, staring:system_call_lab in kernel
# 
```

Problems and Discussions

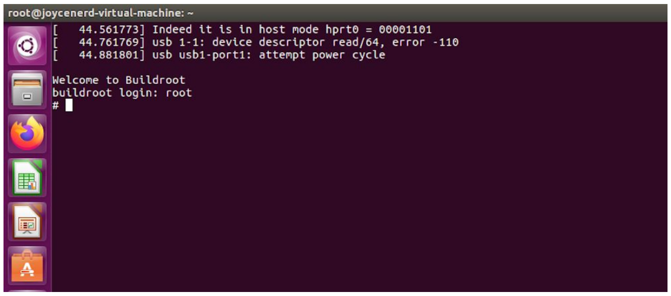
- Q&A
  What is asmlinkage?
  This tells the compiler that the function should not expect to find any of its arguments in registers, but only on the CPU's stack. All system calls are marked with the asmlinkage tag, so they all look to the stack for the arguments.

- Discussions

  The first problem I encountered is that when using extra monitor and keyboard to display the message of Pi, there is an error about the device, which **doesn't allow the keyboard and mouse or any other extra device**. This error affects a lot cause this means we can't log in to the system in Pi. Another classmate found out that when **using USB to TTL serial transmission cable and remote login the system** although it still shows the error, in this way I can log in to the system cause I'm not plugging an extra device into Pi.



```
[    2.360197] random: crng init done
[    2.772477] irq 39: nobody cared (try booting with the "irqpoll" option)
[    2.787285] CPU: 0 PID: 0 Comm: swapper/0 Not tainted 4.9.80-v7+ #1
[    2.801686] Hardware name: Generic DT based system
[    2.814628] [<8010fa48>] (unwind_backtrace) from [<8010c058>] (show_stack+0x20/0x24)
[    2.830539] [<8010c058>] (show_stack) from [<80457a04>] (dump_stack+0xd4/0x118)
[    2.845948] [<80457a04>] (dump_stack) from [<80176ea0>] (__report_bad_irq+0x38/0xd0)
[    2.861840] [<80176ea0>] (__report_bad_irq) from [<801772a4>] (note_interrupt+0x280/0x2d0)
[    2.878352] [<801772a4>] (note_interrupt) from [<801743a0>] (handle_irq_event_percpu+0x5c/0x68)
[    2.895406] [<801743a0>] (handle_irq_event_percpu) from [<80174400>] (handle_irq_event+0x54/0x78)
[    2.912712] [<80174400>] (handle_irq_event) from [<80177b30>] (handle_level_irq+0xb0/0x158)
[    2.929498] [<80177b30>] (handle_level_irq) from [<801732e0>] (generic_handle_irq+0x34/0x44)
[    2.946409] [<801732e0>] (generic_handle_irq) from [<80480180>] (bcm2836_chained_handle_irq+0x38/0x50)
[    2.964315] [<80480180>] (bcm2836_chained_handle_irq) from [<801732e0>] (generic_handle_irq+0x34/0x44)
[    2.982252] [<801732e0>] (generic_handle_irq) from [<8017390c>] (__handle_domain_irq+0x6c/0xc4)
[    2.999648] [<8017390c>] (__handle_domain_irq) from [<8010150c>] (bcm2836_arm_irqchip_handle_irq+0xa8/0xac)
[    3.018213] [<8010150c>] (bcm2836_arm_irqchip_handle_irq) from [<8072753c>] (__irq_svc+0x5c/0x7c)
[    3.035984] Exception stack(0x80c01f18 to 0x80c01f60)
[    3.049948] 1f00:                                                        00000000 b8b8c470
[    3.067130] 1f20: 00000000 00000000 80c00000 80c0312c 00000001 80c03198 80c727bd 80c727bd
[    3.084363] 1f40: b93ffb00 80c01f74 80c040a4 80c01f68 80108a54 80108a58 60000013 ffffffff
[    3.101593] [<8072753c>] (__irq_svc) from [<80108a58>] (arch_cpu_idle+0x34/0x4c)
[    3.118099] [<80108a58>] (arch_cpu_idle) from [<80726c94>] (default_idle_call+0x34/0x48)
[    3.135388] [<80726c94>] (default_idle_call) from [<80162020>] (cpu_startup_entry+0xe4/0x160)
[    3.153220] [<80162020>] (cpu_startup_entry) from [<807214a8>] (rest_init+0x84/0x88)
[    3.170349] [<807214a8>] (rest_init) from [<80b00d34>] (start_kernel+0x3c0/0x3cc)
[    3.187236] [<80b00d34>] (start_kernel) from [<0000807c>] (0x807c)
[    3.202792] handlers:
[    3.214316] [<80576764>] dwc_otg_common_irq
[    3.227761] [<80584950>] dwc_otg_pcd_irq
[    3.240879] [<80562ec4>] usb_hcd_irq
[    3.253583] Disabling IRQ #39
[    3.273902] mmc0: host does not support reading read-only switch, assuming write-enable
[    3.297824] mmc0: new high speed SDHC card at address aaaa
[    3.313388] mmcblk0: mmc0:aaaa SC16G 14.8 GiB
[    3.329082] mmc1: queuing unknown CIS tuple 0x80 (2 bytes)
[    3.343757]  mmcblk0: p1 p2
[    3.357010] mmc1: queuing unknown CIS tuple 0x80 (3 bytes)
[    3.373265] mmc1: queuing unknown CIS tuple 0x80 (3 bytes)
[    3.390514] mmc1: queuing unknown CIS tuple 0x80 (7 bytes)
[    3.435774] EXT4-fs (mmcblk0p2): couldn't mount as ext3 due to feature incompatibilities
[    3.456539] EXT4-fs (mmcblk0p2): couldn't mount as ext2 due to feature incompatibilities
[    3.477957] EXT4-fs (mmcblk0p2): INFO: recovery required on readonly filesystem
[    3.494071] EXT4-fs (mmcblk0p2): write access will be enabled during recovery
[    3.496310] mmc1: new high speed SDIO card at address 0001
[    3.547187] EXT4-fs (mmcblk0p2): recovery complete
[    3.565476] EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode. Opts: (null)
[    3.582593] VFS: Mounted root (ext4 filesystem) readonly on device 179:2.
[    3.602128] devtmpfs: mounted
[    3.615724] Freeing unused kernel memory: 1024K
[    3.706507] EXT4-fs (mmcblk0p2): re-mounted. Opts: data=ordered
[    7.281812] Indeed it is in host mode hprt0 = 00001101
[    7.491778] usb 1-1: device descriptor read/64, error -110

Welcome to Buildroot
buildroot login: [   22.721812] Indeed it is in host mode hprt0 = 00001101
[   22.931777] usb 1-1: device descriptor read/64, error -110
[   23.061827] Indeed it is in host mode hprt0 = 00001101
[   23.271772] usb 1-1: new high-speed USB device number 3 using dwc_otg
[   28.321783] Indeed it is in host mode hprt0 = 00001101
[   28.531770] usb 1-1: device descriptor read/64, error -110
_
```



```
[   44.561773] Indeed it is in host mode hprt0 = 00001101
[   44.761769] usb 1-1: device descriptor read/64, error -110
[   44.881801] usb usb1-port1: attempt power cycle

Welcome to Buildroot
buildroot login: root
# 
```

The second problem is after typing dmesg to see if "Initialize trap" and "arm: system call handler initialization -> see assembly code\n" are shown on the screen. There are so many messages and I can't roll up to see the previous page. Then I find where the dmesg is saved in, which contains the original dmesg in it. It is in **/var/log/message**s. I search for the specific message inside that file and it actually works.