Debugging Methodology Report

김동민 교육생

목차

1. 디버깅 도구

디버깅 도구

printk

커널 로그를 출력하는 커널 함수

- 커널의 printf
- dmesg로 로그 확인
- log level 설정 가능

ftrace

리눅스 커널에서 함수 호출을 추적하고 성능 분석을 수행하는 강력한 트레이싱 도구

- 함수 호출 추적
- 이벤트 추적
- 동적 활성화/비활성화 가능

printk란

printk는 커널 레벨에서 디버깅메시지를 출력할 수 있는 함수이다.

형식 : printk(LOG_LEVEL "출력할 문자열(포맷스트링)", ...);

ex)
```
printk(KERN_INFO "current function : %s\n", __func__);
```

dmesg로 출력가능

```
root@jammy:~# dmesg | grep current
[    1.699465] current function : tcc_dwmac_probe
root@jammy:~#
```

printk

printk()를 사용하면서 로그 레벨을 지정할 수 있다.

```
printk(KERN_INFO "current function : %s\n", __func__);
```

로그레벨:

KERN_EMERG(0)
KERN_ALERT
KERN_CRIT
KERN_ERR
KERN_WARNING
KERN_NOTICE
KERN_INFO
KERN_DEBUG(7)

*숫자가 낮을 수록 우선순위가 높다.

아래의 경로에 콘솔 출력에 대한 레벨을 설정 가능

```
vboxuser@topst:~/autolinux$  cat /proc/sys/kernel/printk
4       4       1       7
vboxuser@topst:~/autolinux$
```

current console level: 현재 콘솔 레벨.
해당 레벨보다 낮은 레벨의 printk는 화면에 바로 출력X

default message level: printk에 콘솔 레벨을 명시하지 않았을 때 지정되는 기본 레벨

minimum console level: current console level에 지정할 수 있는 최소 레벨

default console level: 부팅했을 때 해당 값으로 current console level을 초기화

ftrace란

커널 소스를 수정하지 않고 동적으로 디버깅을 수행할 수 있는 도구

이벤트 추적, 함수 추적 등의 기능을 제공한다.

ftrace 설치

1. menuconfig 실행

./autolinux -c build "linux-telechips -c menuconfig"

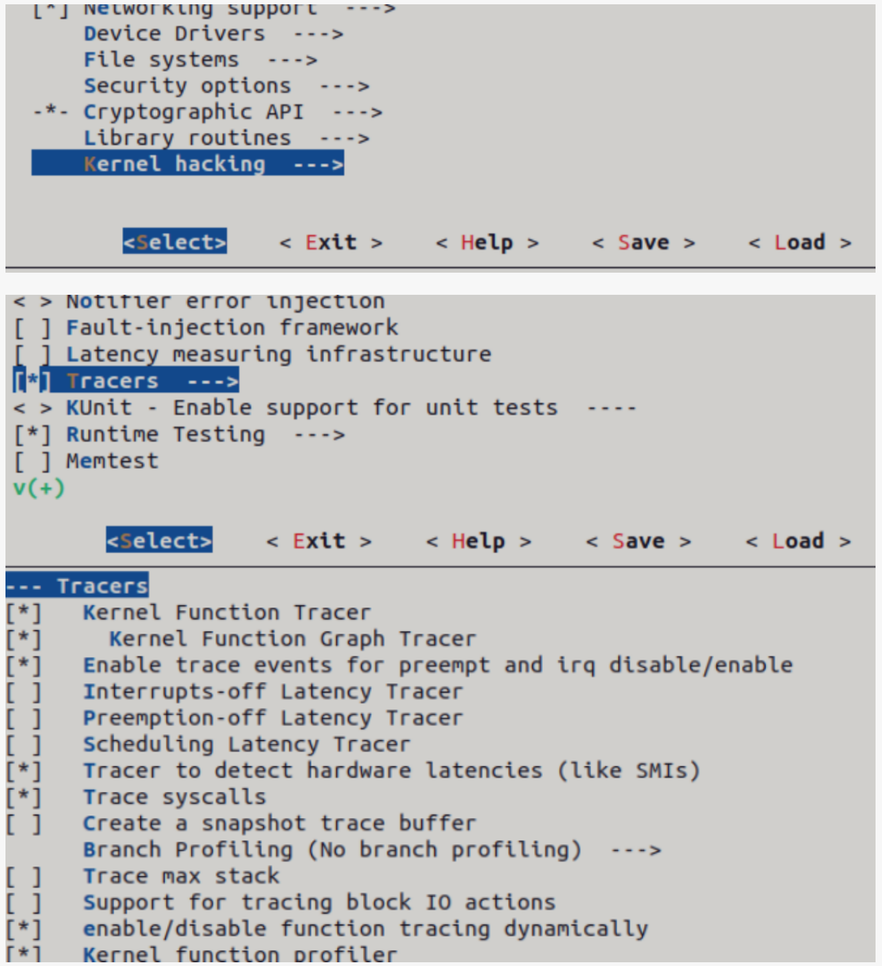
2. kernel hacking > Tracers(built-in 설정) 에서
포함하고 싶은 기능을 built-in으로 설정

ex) Kernel Function Tracer, Trace syscalls등..

3. 저장하여 .config파일 생성

4. 커널 빌드
./autolinux -c build 혹은
./autolinux -c build "linux-telechips -C compile"

5. fwdn으로 TOPST에 이미지 굽기



ftrace 실행(함수 추적)

cd /sys/kernel/debug/tracing (ftrace 관련 디렉터리로 이동)

1. sysctl kernel.ftrace_enabled=1 (ftrace 를 사용 가능하도록 변경)

2. echo function > current_tracer (함수 추적 기능 ON)

3. echo "*stmmac*" > set_ftrace_filter
(추적할 함수명을 세팅)

echo 1 > tracing_on (ftrace 시작)

(데이터가 쌓이도록 잠시 대기...)

4. echo 0 > tracing_on (ftrace 종료)

5. cat trace (결과 확인)

```
#                                _-----=> irqs-off
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| /     delay
#           TASK-PID     CPU#  ||||   TIMESTAMP  FUNCTION
#              | |         |   ||||      |         |
     kworker/2:2-612     [002] ...1  3180.387861: stmmac_mdio_read <-__mdiobus_read
     kworker/2:2-612     [002] ...1  3180.387976: stmmac_mdio_read <-__mdiobus_read
     kworker/2:2-612     [002] ...1  3181.411854: stmmac_mdio_read <-__mdiobus_read
     kworker/2:2-612     [002] ...1  3181.411970: stmmac_mdio_read <-__mdiobus_read
     kworker/2:2-612     [002] ...1  3182.435851: stmmac_mdio_read <-__mdiobus_read
     kworker/2:2-612     [002] ...1  3182.435965: stmmac_mdio_read <-__mdiobus_read
     kworker/2:2-612     [002] ...1  3183.459857: stmmac_mdio_read <-__mdiobus_read
     kworker/2:2-612     [002] ...1  3183.459970: stmmac_mdio_read <-__mdiobus_read
           ping-672      [001] ...3  3183.985023: stmmac_xmit <-netdev_start_xmit
           ping-672      [001] ...3  3183.985026: stmmac_vlan_insert <-stmmac_xmit
           ping-672      [001] ...3  3183.985034: stmmac_tx_timer_arm <-stmmac_xmit
    rb_consumer-124      [000] d.h2  3183.985038: stmmac_interrupt <-__handle_irq_event_percpu
    rb_consumer-124      [000] ..s2  3183.985045: stmmac_napi_poll_tx <-net_rx_action
    rb_consumer-124      [000] d.h1  3183.986468: stmmac_interrupt <-__handle_irq_event_percpu
    rb_consumer-124      [000] ..s1  3183.986472: stmmac_napi_poll_rx <-net_rx_action
    rb_consumer-124      [000] ..s1  3183.986508: stmmac_napi_poll_tx <-net_rx_action
         <idle>-0        [001] ..s2  3183.991841: stmmac_tx_timer <-call_timer_fn
         <idle>-0        [001] ..s2  3183.991852: stmmac_napi_poll_tx <-net_rx_action
     kworker/2:2-612     [002] ...1  3184.483864: stmmac_mdio_read <-__mdiobus_read
     kworker/2:2-612     [002] ...1  3184.483976: stmmac_mdio_read <-__mdiobus_read
```

2. 에러 발생 및 디버깅

에러발생 및 디버깅

문제 상황 : ping은 정상적이지만 대용량 데이터(10G)를 송신하려고 하면

"Tx Ring full when queue awake" 이라는 에러메시지가 출력

->Tx ring에 문제발생

```
PING 10.42.0.2 (10.42.0.2) 56(84) bytes of data.
64 bytes from 10.42.0.2: icmp_seq=1 ttl=128 time=1.50 ms
64 bytes from 10.42.0.2: icmp_seq=2 ttl=128 time=1.50 ms
64 bytes from 10.42.0.2: icmp_seq=3 ttl=128 time=1.48 ms
64 bytes from 10.42.0.2: icmp_seq=4 ttl=128 time=1.31 ms
^C
--- 10.42.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 1.312/1.449/1.503/0.079 ms
root@jammy:~# iperf3 -c 10.42.0.2 -n 10G
Connecting to host 10.42.0.2, port 5201
[  5] local 10.42.0.3 port 53622 connected to 10.42.0.2 port 5201
[  165.261558] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
[  165.269432] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
[  165.278415] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
[  165.287085] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
[  165.295782] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
[  165.303622] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
[  165.312226] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
[  165.320022] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
[  165.329289] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
[  165.337481] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
[  165.346124] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
```

에러 발생 및 디버깅

1. 송신할 때 어떤일이 발생하는지 확인하기 위해 네트워크 인터페이스의 msglvl을 변경

-> ethtool -s eth0 msglvl 0x1000

```
root@jammy:~# ethtool -s eth0 msglvl 0x1000
root@jammy:~#
```

```
enum {
        NETIF_MSG_DRV           = 0x0001,
        NETIF_MSG_PROBE         = 0x0002,
        NETIF_MSG_LINK          = 0x0004,
        NETIF_MSG_TIMER         = 0x0008,
        NETIF_MSG_IFDOWN        = 0x0010,
        NETIF_MSG_IFUP          = 0x0020,
        NETIF_MSG_RX_ERR        = 0x0040,
        NETIF_MSG_TX_ERR        = 0x0080,
        NETIF_MSG_TX_QUEUED     = 0x0100,
        NETIF_MSG_INTR          = 0x0200,
        NETIF_MSG_TX_DONE       = 0x0400,
        NETIF_MSG_RX_STATUS     = 0x0800,
        NETIF_MSG_PKTDATA       = 0x1000,
        NETIF_MSG_HW            = 0x2000,
        NETIF_MSG_WOL           = 0x4000,
};
```

에러 발생 및 디버깅

2. dmesg로 메시지를 확인해보니 tx ring의 상태가 보인다.

tx ring의 descriptor의 개수가 4개뿐이다.

```
root@jammy:~# ethtool -s eth0 msglvl 0x1000
root@jammy:~# dmesg
[ 3415.354132] [INFO][tcc-rtc] set time 2024/12/10 11:09:15
[ 3415.354150] [INFO][tcc-rtc] read time 2024/12/10 11:09:15
[ 3453.423820] TX descriptor ring:
[ 3453.423826] 000 [0x22ae5000]: 0x0 0x0 0x0 0x0
[ 3453.423829] 001 [0x22ae5010]: 0x0 0x0 0x0 0x0
[ 3453.423832] 002 [0x22ae5020]: 0x0 0x0 0x0 0x0
[ 3453.423835] 003 [0x22ae5030]: 0x0 0x0 0x80000000 0x0
[ 3468.423717] TX descriptor ring:
[ 3468.423721] 000 [0x22ae5000]: 0x0 0x0 0x80000000 0x0
[ 3468.423724] 001 [0x22ae5010]: 0x0 0x0 0x0 0x0
[ 3468.423727] 002 [0x22ae5020]: 0x0 0x0 0x0 0x0
[ 3468.423730] 003 [0x22ae5030]: 0x0 0x0 0x0 0x0
[ 3483.423611] TX descriptor ring:
[ 3483.423616] 000 [0x22ae5000]: 0x0 0x0 0x0 0x0
[ 3483.423619] 001 [0x22ae5010]: 0x0 0x0 0x80000000 0x0
[ 3483.423621] 002 [0x22ae5020]: 0x0 0x0 0x0 0x0
[ 3483.423624] 003 [0x22ae5030]: 0x0 0x0 0x0 0x0
[ 3498.423530] TX descriptor ring:
[ 3498.423534] 000 [0x22ae5000]: 0x0 0x0 0x0 0x0
[ 3498.423537] 001 [0x22ae5010]: 0x0 0x0 0x0 0x0
[ 3498.423540] 002 [0x22ae5020]: 0x0 0x0 0x80000000 0x0
[ 3498.423542] 003 [0x22ae5030]: 0x0 0x0 0x0 0x0
```

에러 발생 및 디버깅

3. alloc_dma_tx_desc_resources에서 tx ring을 할당하는 코드로 이동

-> DMA_TX_SIZE가 tx ring의 개수라고 추측

```
tx_q->dma_tx = dma_alloc_coherent(priv->device,
                  DMA_TX_SIZE * sizeof(struct dma_desc),
                  &tx_q->dma_tx_phy,
                  GFP_KERNEL);
if (!tx_q->dma_tx)
    goto err_dma;
```

에러 발생 및 디버깅

4. #define DMA_TX_SIZE 4

-> descriptor를 4개만 생성했기에 대용량 데이터 송신에 사용할 descriptor가 부족

DMA_TX_SIZE 512로 수정해준다.

```
35 #define DWXGMAC_CORE_2_10   0x21
36
37 #define STMMAC_CHAN0    0   /* Always supported and default f
38
39 /* These need to be power of two, and >= 4 */
40 #define DMA_TX_SIZE 4
41 #define DMA_RX_SIZE 512
42 #define STMMAC_GET_ENTRY(x, size)   ((x + 1) & (size - 1))
43
44 #undef FRAME_FILTER_DEBUG
45 /* #define FRAME_FILTER_DEBUG */
46
```

수정

```
38
39 /* These need to be power of two, and >= 4 */
40 #define DMA_TX_SIZE 512
41 #define DMA_RX_SIZE 512
42 #define STMMAC_GET_ENTRY(x, size)   ((x + 1) & (size - 1))
43
44 #undef FRAME_FILTER_DEBUG
45 /* #define FRAME_FILTER_DEBUG */
```

에러 발생 및 디버깅

```
PING 10.42.0.2 (10.42.0.2) 56(84) bytes of data.
64 bytes from 10.42.0.2: icmp_seq=1 ttl=128 time=1.50 ms
64 bytes from 10.42.0.2: icmp_seq=2 ttl=128 time=1.50 ms
64 bytes from 10.42.0.2: icmp_seq=3 ttl=128 time=1.48 ms
64 bytes from 10.42.0.2: icmp_seq=4 ttl=128 time=1.31 ms
^C
--- 10.42.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 1.312/1.449/1.503/0.079 ms
root@jammy:~# iperf3 -c 10.42.0.2 -n 10G
Connecting to host 10.42.0.2, port 5201
[  5] local 10.42.0.3 port 53622 connected to 10.42.0.2 port 5201
[  165.261558] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
[  165.269432] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
[  165.278415] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
[  165.287085] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
[  165.295782] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
[  165.303622] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
[  165.312226] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
[  165.320022] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
[  165.329289] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
[  165.337481] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
[  165.346124] tcc-dwmac 11c00000.gmac eth0: stmmac_xmit: Tx Ring full when queue awake
```

디버깅

```
root@jammy:~# ping 10.42.0.2
PING 10.42.0.2 (10.42.0.2) 56(84) bytes of data.
64 bytes from 10.42.0.2: icmp_seq=1 ttl=128 time=0.269 ms
64 bytes from 10.42.0.2: icmp_seq=2 ttl=128 time=0.438 ms
^C
--- 10.42.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1017ms
rtt min/avg/max/mdev = 0.269/0.353/0.438/0.084 ms
root@jammy:~# iperf3 -c 10.42.0.2 -n 10G
Connecting to host 10.42.0.2, port 5201
[  5] local 10.42.0.3 port 35370 connected to 10.42.0.2 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  77.7 MBytes   652 Mbits/sec   85    312 KBytes
[  5]   1.00-2.00   sec   113 MBytes   951 Mbits/sec    0    331 KBytes
[  5]   2.00-3.00   sec   113 MBytes   947 Mbits/sec    0    336 KBytes
[  5]   3.00-4.00   sec   113 MBytes   952 Mbits/sec    0    361 KBytes
[  5]   4.00-5.00   sec   112 MBytes   938 Mbits/sec    0    371 KBytes
[  5]   5.00-6.00   sec   113 MBytes   949 Mbits/sec    0    371 KBytes
[  5]   6.00-7.00   sec   113 MBytes   951 Mbits/sec    0    374 KBytes
[  5]   7.00-8.00   sec   113 MBytes   947 Mbits/sec    0    374 KBytes
```

5. iperf3로 대용량 데이터가 정상적으로 송신됨을 확인할 수 있다.

디버깅 과정 요약

에러 : ping은 정상적이나 대용량 데이터 송신 불가

1. "Tx Ring full when queue awake" 에러메시지를 통해 tx ring에 문제가 있음을 확인

2. 송신 관련 디버깅 메시지를 보기 위해 ethtool로 메시지 활성화

3. dmesg 를 통해 tx ring에 descriptor의 수가 부족함을 확인

4. DMA_TX_SIZE 값이 작다는 것을 확인하고 크기를 원래대로 복구

5. 커널 빌드 후 TOPST에 굽고, iperf3로 대용량 데이터 전송 -> 정상적으로 송신

감사합니다.