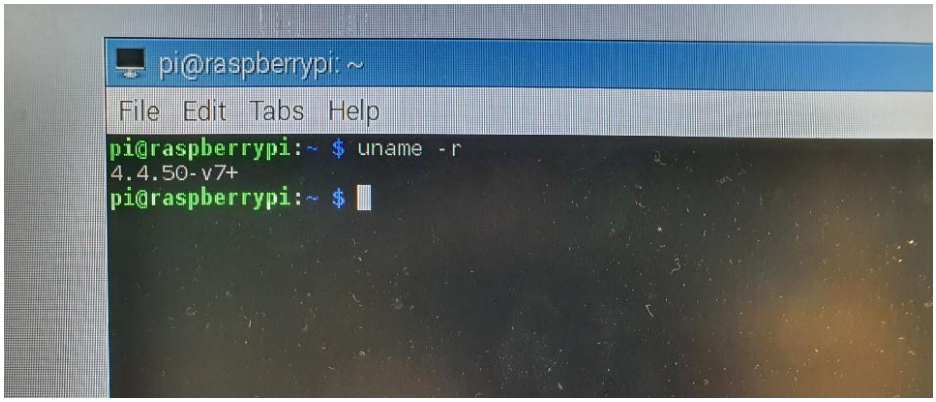
IoTSW_15_최낙관(2016101124)

- System call

1. 실습 예제

이번 실습에서는 System call을 등록, 사용하는 법을 실습할 수 있었습니다.

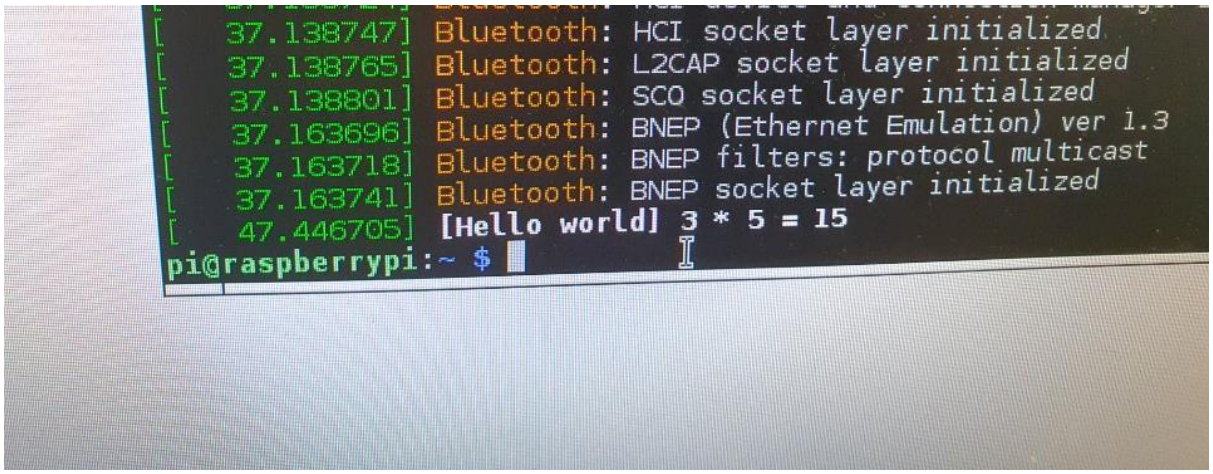
먼저 kernel update 부분에서는 working 폴더에 kernel을 다운받아 적혀 있는 대로 실습을 진행했습니다. 실습은 문제없이 진행됐지만 라즈베리 파이에는 bcm2837칩이 들어가는 것으로 적혀 있었는데, 찾아본 바로는 bcm2937은 bcm2710 package를 쓴다고 나와있었습니다. Chip에 따라 Compile 환경 설정을 하기 위해 해당 코드를 넣어주는 것으로 알고 있었는데 자세한 차이를 찾기 힘들었습니다. 또한 -j 옵션은 단순히 compile시 쓸 CPU core의 개수라고 알고 있는데, 숫자를 낮게 했을 때 compile이 잘되는 문제가 있었습니다. 속도만 늦어질 것이라 예상했는데, no child process라는 오류가 뜨면서 compile이 되지 않는 문제가 있었지만 이유는 찾지 못했습니다.



해당 작업들을 완료하고, raspberry pi에 옮겼을 때 kernel version이 업데이트 된 것을 확인할 수 있었습니다.

System call 함수 추가 또한 system call 번호를 등록하고, system call table에 함수를 등록한 후, 선언 구현을 통해, kernel에 system call을 등록할 수 있었습니다.

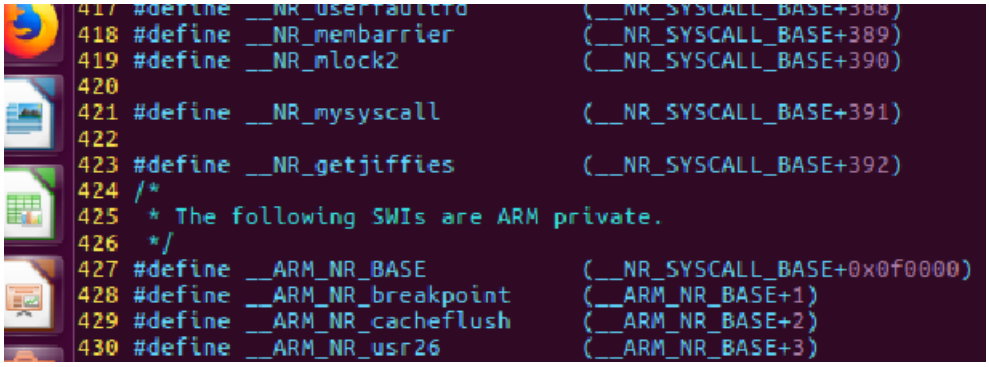
해당 system call을 좀 더 편리하게 쓰기 위해, library를 만들고, 구동 app을 예제를 따라 만들어,



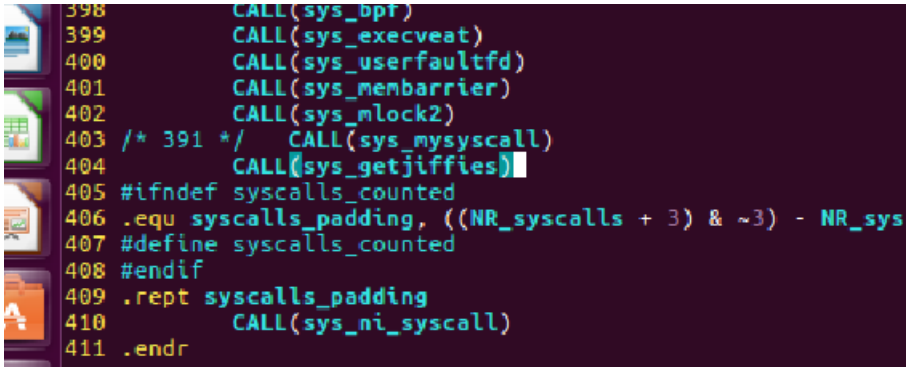
해당 system call의 등록과 작동을 확인할 수 있었습니다.

2. 실습 과제

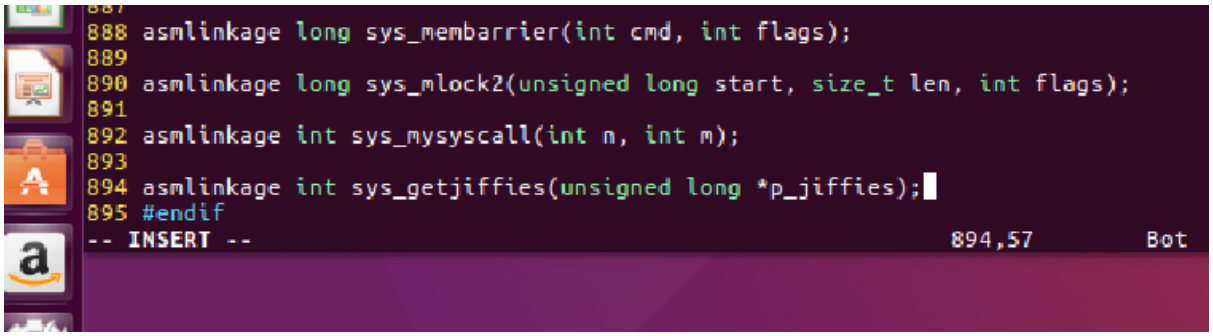
실습 과제는 system call을 등록하는 과제였습니다. 따라서



System call에 번호를 부여하고,



해당 함수를 system call table에 등록한 후,



함수 선언,

nakkwan@nakkwan-VirtualBox: ~/working/linux

```
#include <linux/unistd.h>
#include <linux/errno.h>
#include <linux/sched.h>
#include <linux/jiffies.h>
#include <asm/uaccess.h>

asmlinkage int sys_getjiffies(unsigned long *p_jiffies){
    unsigned long jiff = (unsigned long)get_jiffies_64();
    copy_to_user(p_jiffies, &jiff, sizeof(jiff));
    printk("[jiffies]: %lu\n", jiff);
    return 0;
}
```

정의를 통해, kernel에 system call을 등록해 주었습니다. User 영역에서 kernel 영역에 바로 접근할 수 없기 때문에 copy_to_user()함수를 사용하여 해당 data를 user 영역으로 넘겨주었습니다.

Library와 application작성은 실습 예제와 마찬가지로,

Terminal

nakkwan@nakkwan-VirtualBox: ~/working/syscall

```
#include "/home/nakkwan/working/linux/arch/arm/include/uapi/asm/unistd.h"

int getjiffies(unsigned long *p_jiffies){
    return syscall(__NR_getjiffies, p_jiffies);
}
```

nakkwan@nakkwan-VirtualBox: ~/working/syscall

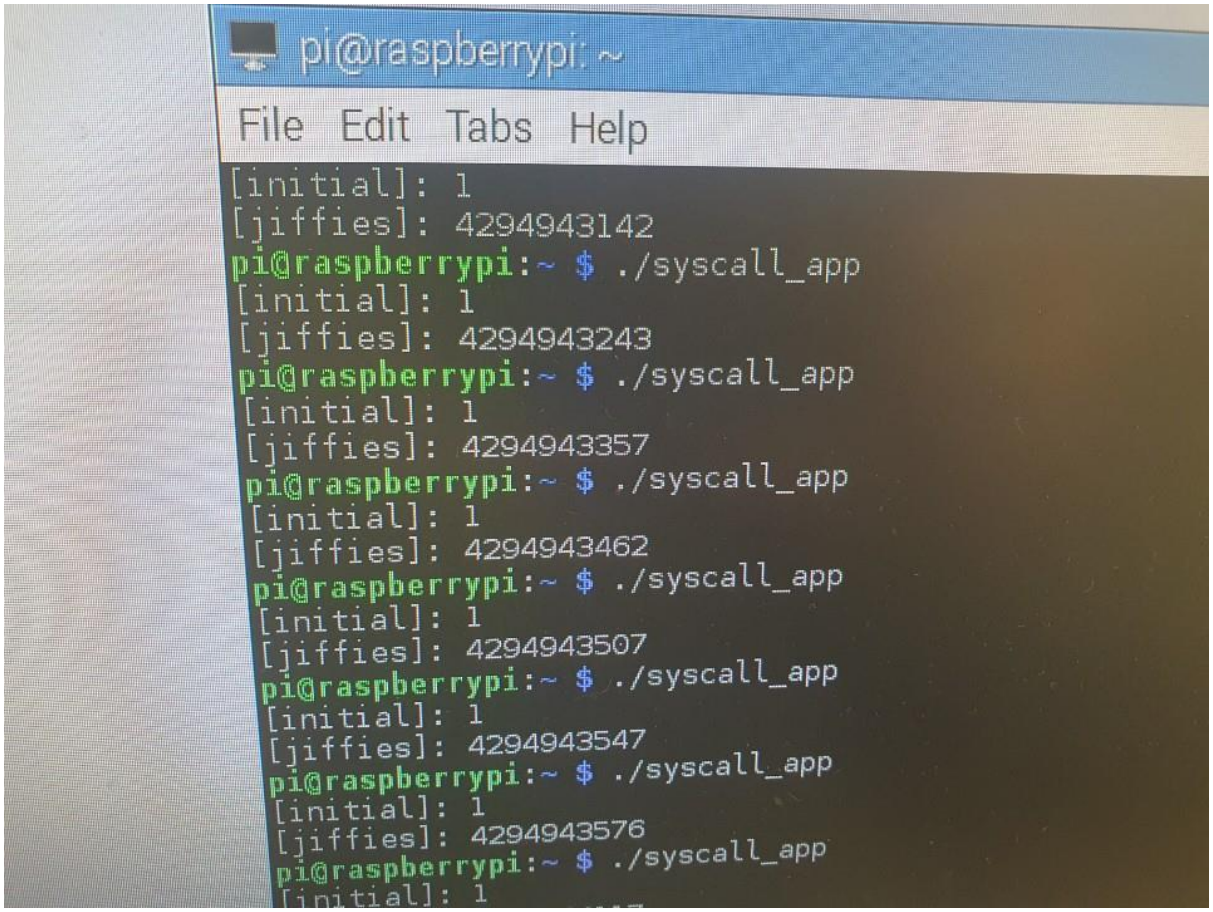
```
#include <stdio.h>

int main(){
    unsigned long p = 1;
    printf("[initial]: %lu\n", p);

    getjiffies(&p);
    printf("[jiffies]: %lu\n", p);

    return 0;
}
```

위와 같이 작성했습니다. 기존에 넣어준 값과 비교하기 위해, p값을 먼저 출력한 후, jiffies값을 넣어주었습니다.



해당 system call의 출력이 잘 되는 것을 확인할 수 있었습니다. Raspberry pi는 기본으로 100Hz이기 때문에, 해당하는 시간마다 jiffies가 높아지는 것을 확인할 수 있었습니다.