

2019년 2학기 운영체제

Assignment 1

System Software Laboratory

School of Computer and Information Engineering

Kwangwoon Univ.

Requirements

- **Report**

- Introduction : 5줄 이하
 - **Background 제외**
- Reference : 각 과제 별
 - e.g. 친구 도움, 책, 인터넷 사이트 주소 등
 - 강의자료만 이용한 경우 생략 가능
- Conclusion
 - **분석 결과 포함**

- **Source**

- [Assignment 1-2]
 - 수정한 코드를 첨부하고 보고서에 수정한 코드의 path를 명시할 것
- [Assignment 1-3]
 - add.c, test.c, my_wrapper.c


- **Copy 발견 시 0점 처리**

Requirements (cont'd)

▪ Softcopy

- 보고서 및 소스파일은 하나의 압축파일로 압축하여 제출 (**tar.gz**)
- 보고서 및 압축 파일 명은 **os_과제번호_학번** 으로 수정

- e.g.

	(보고서)	(압축 파일)
	OS_1_2017202001.pdf	OS_1_2017202001.tar.gz
-  보고서는 "PDF"로 변환하여 제출

- **2019. 10. 14(월) 23:59:59** 까지, [U-캠퍼스] → [온라인참여학습] → [과제제출]
- Delay 없음
- 업로드 양식에 어긋날 경우 감점 처리

▪ Hardcopy

- 제출하지 않음
- 과제 마감 당일 (**10/14 월**) 오후 5시 이전까지 발송한 질문에 한하여 답변
 - 남건욱 (심동규 교수님 연구실 / ngotic@kw.ac.kr)
 - ✓ 위의 메일로 온 질문에만 답변합니다.

Assignment 1-1

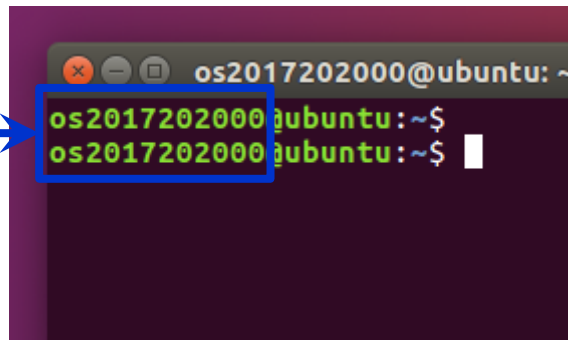
- Linux Installation & kernel compile

- Linux Installation

- 리눅스를 설치하는 과정을 캡처하고, 이를 짧은 설명과 함께 보고서에 첨부

- 과제 요구 사항

- 리눅스 설치 과정을 간단하게 캡처 및 설명
 - 가상 머신에서 설치하는 경우, 가상 머신을 만드는 과정도 포함
 - 계정 ID는 “os_학번” 또는 “os학번”으로 할 것
 - (e.g os_2017202000 또는 os2017202000)



Assignment 1-1

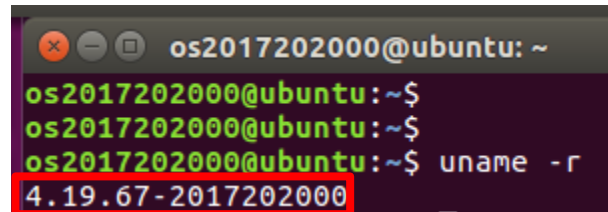
- Linux Installation & kernel compile

- Kernel compile

- kernel을 compile하는 과정을 캡처하고, 이를 짧은 설명과 함께 보고서에 첨부

- 과제 요구 사항

- Kernel version은 4.19.67로 할 것
 - 컴파일 과정을 terminal과 vi를 사용하여 캡처
 - 각 명령어가 어떠한 기능을 하였는지 간단히 서술
 - 컴파일 과정이 끝난 후 재 부팅 후 버전 확인
 - “uname -r” 의 결과가 “4.19.67-본인 학번” 이 나오도록 진행할 것

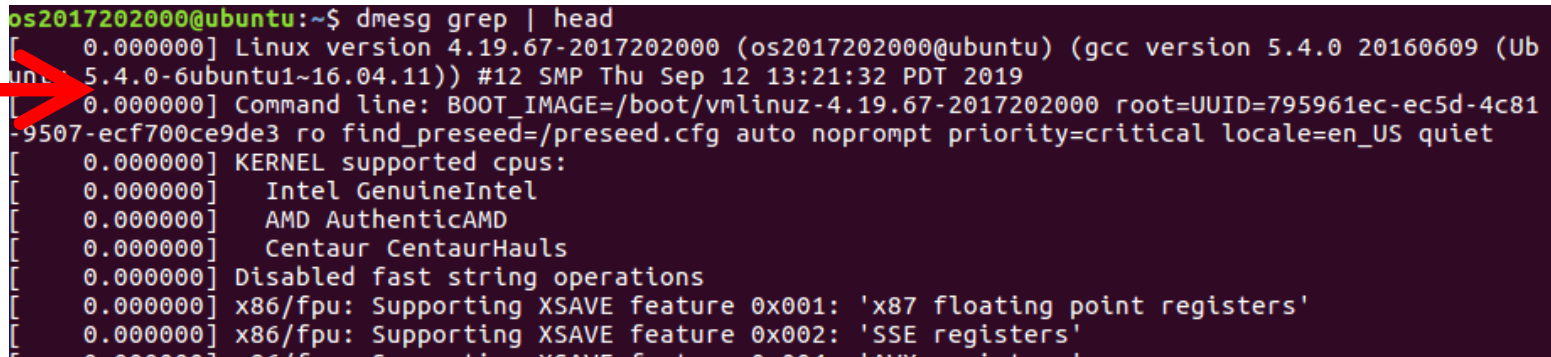


A terminal window screenshot with a dark background. The title bar shows 'os2017202000@ubuntu: ~'. The terminal content shows three lines of text: 'os2017202000@ubuntu:~\$', 'os2017202000@ubuntu:~\$', and 'os2017202000@ubuntu:~\$ uname -r'. The output of the command, '4.19.67-2017202000', is displayed on the fourth line and is highlighted with a red rectangular box.

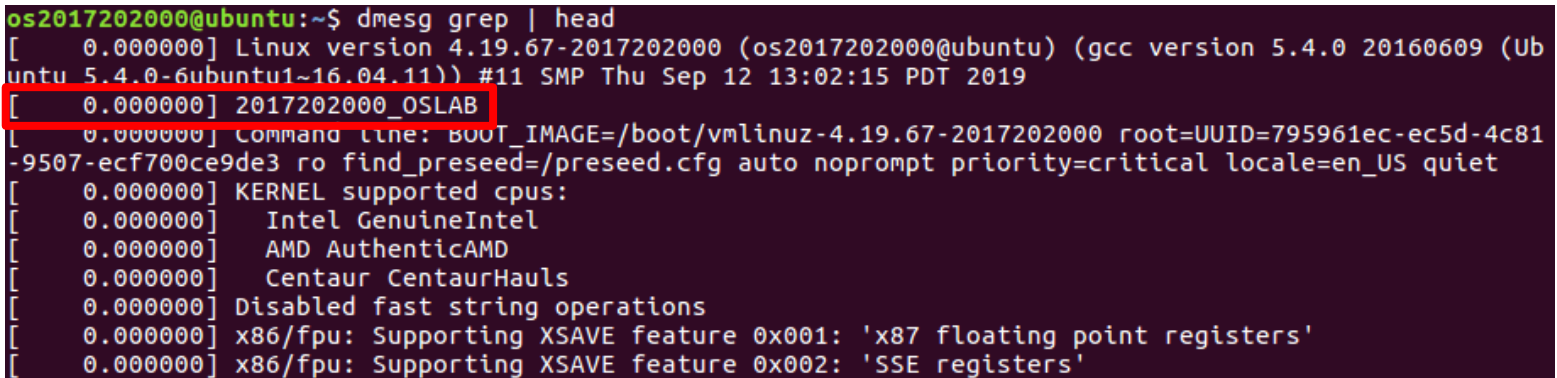
Assignment 1-2

■ 커널 코드 수정

- dmesg로 kernel message를 확인한 후, 아래의 그림에서 화살표가 가리키는 위치에 본인의 학번이 찍히도록 커널 코드 수정
- 출력할 메시지: 학번_OSLAB



```
os2017202000@ubuntu:~$ dmesg grep | head
[ 0.000000] Linux version 4.19.67-2017202000 (os2017202000@ubuntu) (gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.11)) #12 SMP Thu Sep 12 13:21:32 PDT 2019
[ 0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.19.67-2017202000 root=UUID=795961ec-ec5d-4c81-9507-ecf700ce9de3 ro find_preseed=/preseed.cfg auto noprompt priority=critical locale=en_US quiet
[ 0.000000] KERNEL supported cpus:
[ 0.000000]   Intel GenuineIntel
[ 0.000000]   AMD AuthenticAMD
[ 0.000000]   Centaur CentaurHauls
[ 0.000000] Disabled fast string operations
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers'
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
```



```
os2017202000@ubuntu:~$ dmesg grep | head
[ 0.000000] Linux version 4.19.67-2017202000 (os2017202000@ubuntu) (gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.11)) #11 SMP Thu Sep 12 13:02:15 PDT 2019
[ 0.000000] 2017202000_OSLAB
[ 0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.19.67-2017202000 root=UUID=795961ec-ec5d-4c81-9507-ecf700ce9de3 ro find_preseed=/preseed.cfg auto noprompt priority=critical locale=en_US quiet
[ 0.000000] KERNEL supported cpus:
[ 0.000000]   Intel GenuineIntel
[ 0.000000]   AMD AuthenticAMD
[ 0.000000]   Centaur CentaurHauls
[ 0.000000] Disabled fast string operations
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers'
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
```

Assignment 1-2

- 커널 코드 수정

- 과제 요구사항

- 커널 코드 내에 printk() 함수를 사용한 코드를 추가하여 과제 수행
 - 로그 레벨은 **KERN_INFO** 사용 (“Appendix A” 참고)
 - 부팅 직후 \$ dmesg 으로 확인

- 보고서에 다음의 내용을 필히 포함

- 리눅스 커널 코드에서 수정한 부분을 명시 (소스 코드 path도 포함)
 - ex. init/main.c를 수정했다면
 - main.c에서 수정한 부분과 수정한 수정코드 path를 보고서에 명시할 것
 - 수정한 위치가 정당한 이유를 명시할 것
 - 결과 화면
 - Softcopy 제출 시, 수정한 커널 코드 파일 포함

- Hints

- cscope, ctags 등을 활용할 것

Assignment 1-3

- **System call wrapping**
 - **System call 작성**
 - a와 b를 더하는 system call “sys_add” 작성
 - 구현 파일의 이름은 “add.c”로 할 것.
 - sys_add **시스템 콜 번호는 549**를 사용할 것
 - **Module 작성**
 - sys_add를 wrapping 할 것
 - 이 경우, “sys_add” 호출 시 곱셈을 수행
 - ex. sys_add(5, 3) = 15
 - Module 적재 및 제거했을 때 커널 메시지 확인 (\$ dmesg)
 - 11페이지 예시 결과 참고
 - 구현 파일의 이름은 “my_wrapper.c”로 할 것.

Assignment 1-3

- **System call wrapping**
 - 테스트 파일 작성
 - 테스트 파일명: "test.c"
 - 코드 실행 시 연산을 수행할 정수 2개를 입력 (ex \$./test 5 3)
 - 두 정수를 다음과 같이 연산하는 코드 작성
 - System call wrapping 전, 임의의 두 숫자를 더하는 코드
 - System call wrapping 후, 임의의 두 숫자를 곱하는 코드
 - 두 숫자의 연산 결과를 출력할 것
 - 단, 연산은 본인이 작성한 **system call을 이용**하여 행할 것

Assignment 1-3

- **System call wrapping**

- **과제 요구사항**

- System call, module, 테스트 파일 작성 과정을 보고서에 캡처 및 설명할 것
 - 커널 메시지 출력
 - Module 적재 시 : 커널 메시지로 "init_[본인 학번]"을 출력
 - Module 제거 시 : 커널 메시지로 "exit_[본인 학번]"을 출력
 - ex.
 - Module 적재 시 (\$ dmesg)

```
root@ubuntu:/home/os2017202000/Downloads/result# insmod my_wrapper.ko
root@ubuntu:/home/os2017202000/Downloads/result# dmesg | tail -n 1
[ 47.473684] init_[2017202000]
```

- Module 제거 시 (\$ dmesg)

```
root@ubuntu:/home/os2017202000/Downloads/result# rmmod my_wrapper
root@ubuntu:/home/os2017202000/Downloads/result# dmesg | tail -n 1
[ 64.295027] exit_[2017202000]
```

Assignment 1-3

- System call wrapping

- 결과 예시

```
root@ubuntu:/home/os2017202000/Downloads/result# ./test 5 3
5 op 3 = 8
root@ubuntu:/home/os2017202000/Downloads/result# make
make -C /lib/modules/4.19.67-2017202000/build M=/home/os2017202000/Downloads/res
ult modules
make[1]: Entering directory '/home/os2017202000/Downloads/linux-4.19.67'
Building modules, stage 2.
MODPOST 1 modules
make[1]: Leaving directory '/home/os2017202000/Downloads/linux-4.19.67'
root@ubuntu:/home/os2017202000/Downloads/result# insmod my_wrapper.ko
root@ubuntu:/home/os2017202000/Downloads/result# dmesg | tail -n 1
[ 1311.232568] init_[2017202000]
root@ubuntu:/home/os2017202000/Downloads/result# ./test 5 3
5 op 3 = 15
root@ubuntu:/home/os2017202000/Downloads/result# rmmod my_wrapper
root@ubuntu:/home/os2017202000/Downloads/result# dmesg | tail -n 1
[ 1323.508198] exit_[2017202000]
root@ubuntu:/home/os2017202000/Downloads/result# ./test 5 3
5 op 3 = 8
root@ubuntu:/home/os2017202000/Downloads/result#
```

Appendix A. printk()

- **int printk(const char *fmt, ...);**
 - 커널에서 메시지를 출력하는 함수
 - 로그 레벨 (declared in <linux/kernel.h>)
 - 로그 레벨에 대한 문자열을 상수로 선언해 놓음

상수	문자열	의미
KERN_EMERG	"<0>"	System is unusable
KERN_ALERT	"<1>"	Action must be taken immediately
KERN_CRIT	"<2>"	Critical conditions
KERN_ERR	"<3>"	Error conditions
KERN_WARNING	"<4>"	Warning conditions
KERN_NOTICE	"<5>"	Normal but significant condition
KERN_INFO	"<6>"	Informational
KERN_DEBUG	"<7>"	Debug-level messages

- e.g. **printk(KERN_INFO "just info\n")**
- 사용법은 printf와 유사
- printk()로 출력된 메시지는 \$ dmesg 명령어를 통해 확인

Appendix B. dmesg

- Linux command to print or control the kernel ring buffer

- printk()로 출력한 내용을 본 명령어를 통해 확인할 수 있음

- Usage

- 앞의 5개의 메시지만 보고 싶을 때,

- \$ **dmesg | head -n 5**

```
root@ubuntu:/home/os2017202000/Downloads/result# dmesg | head -n 5
[ 0.000000] Linux version 4.19.67-2017202000 (os2017202000@ubuntu) (gcc versi
on 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.11)) #12 SMP Thu Sep 12 13:21:32
PDT 2019
[ 0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.19.67-2017202000 root=UU
ID=795961ec-ec5d-4c81-9507-ecf700ce9de3 ro find_preseed=/preseed.cfg auto noprom
pt priority=critical locale=en_US quiet
[ 0.000000] KERNEL supported cpus:
[ 0.000000] Intel GenuineIntel
[ 0.000000] AMD AuthenticAMD
```

- 뒤의 5개의 메시지만 보고 싶을 때,

- \$ **dmesg | tail -n 5**

```
root@ubuntu:/home/os2017202000/Downloads/result# dmesg | tail -n 5
[ 7.765653] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
None
[ 7.767153] IPv6: ADDRCONF(NETDEV_CHANGE): ens33: link becomes ready
[ 9.239200] Bluetooth: RFCOMM TTY layer initialized
[ 9.239203] Bluetooth: RFCOMM socket layer initialized
[ 9.239207] Bluetooth: RFCOMM ver 1.11
```

Appendix B. dmesg

- Usage

- 특정 문자가 포함된 메시지 열을 보고 싶을 때

- `$ dmesg | grep XXX`

```
sslab@sslab:~$ dmesg |grep SHY  
[    0.000000] OS_lecture_SHY
```

- 특정 문자가 포함된 메시지 열이 몇 번째 열인지 보고 싶을 때

- `$ dmesg | grep XXX -n`

```
sslab@sslab:~$ dmesg |grep SHY -n  
388:[    0.000000] OS_lecture_SHY
```