과제 1 \_ Kboard v.01

2019년 1학기 운영체제 3분반

201716422 이명규

\_ **서론**

목차

1. **Kboard\_copy, Kboard\_paste** 
   1. 수행결과
   2. 핵심 소스코드
   3. 어려웠던 점 및 해결방안
2. **동기화 문제 해결**

2-2. 동기화 문제 탐지

2-3. 동기화 문제 해결 및 수행결과

2-4. 어려웠던 점

**과제 개요**

kboard system을 구현해, 유저가 copy, paste 기능을 사용할 수 있도록 커널 소스코드와 유저 소스코드를 구현한다. 이 때, 공유 데이터로의 접근으로 인한 동기화 문제를 발견하고 해결한다.

1. **Kboard Copy & Paste**

1 –1. 수행결과

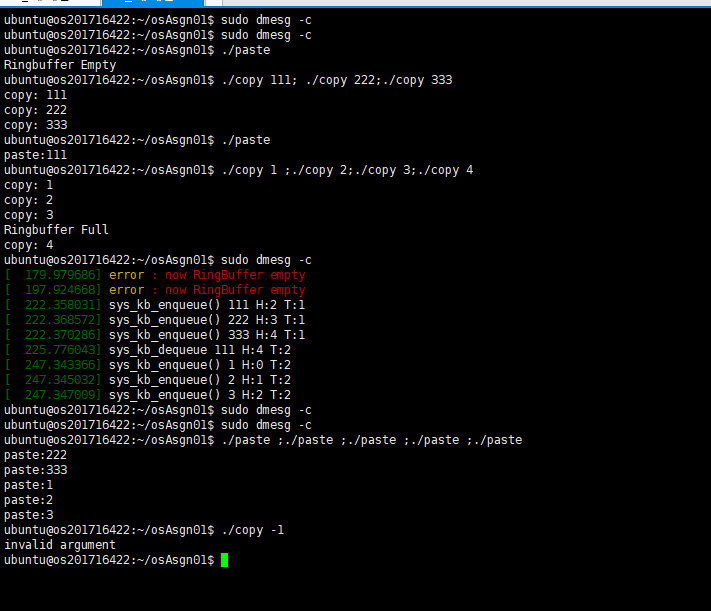


그림 1. 복사/ 붙여넣기

복사 / 붙여넣기 테스트입니다. 과제 안내에 명시된 대로 테스트를 수행하였습니다.

디버그 메시지 초기화 > paste 1회 > copy 3회 > paste 1회 > copy 4회 > 디버그메시지 > paste 5회 > copy -1

그 결과, 버퍼가 가득 찬 경우와 빈 경우에 각각의 에러처리가 된 점을 확인할 수 있었습니다.

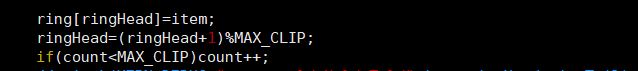
1-2. 주요 소스코드

그림 2. 커널 주요 소스코드

Kboard Copy의 주요 소스코드입니다.

링버퍼 구현사항의 일부로서, 링버퍼 ring의 top에는 ringHead변수가 있으며, Copy 시에 아이템을 저장하고 해당 ringHead체크포인트를 증가시켜주는 역할을 합니다.

count 변수에서는 ring buffer에 저장된 아이템의 개수를 세어주며 이 값을 통해 ring buffer의 full, empty를 판단합니다.

유저영역 Copy의 주요 소스코드입니다.

처음으로 유저가 입력한 인자값을 int형으로 파싱해준 후, kboard\_copy를 통해, syscall에 등록된 do\_sys\_kb\_enqueue를 수행하며, 이 때 반환되는 return값으로 에러처리를 해줍니다.

그림 3. 유저 주요 소스코드

1-3. 어려웠던 점 및 해결방안

1. .작업폴더/kernel에서 Makefile 수정 후, 커널 컴파일 시, 작업폴더에서 Make 하지 않고,

작업폴더/kernel디렉터리에서 make 해서 오류 발생 >> 작업폴더 전환해서 해결!

1. syscall\_table 수정 시, 입력포멧에 맞추지 않고 스페이스바 사용해서 시스템콜 등록되지 않음. (function not implemented 에러메시지) > tab으로 수정해서 해결!
2. ringbuffer 구현 시, 크기 5의 배열을 사용하면 경계문제때문에 4만큼밖에 사용하지 못하는 문제 > 변수추가로 해결
3. **동기화 문제 해결**

2-2. 동기화 문제 탐지

동기화 문제를 탐지할 때, 아래 두 개 코드를 각각 쉘에서 수행해, kboard\_copy와 kboard\_paste의 race condition을 발생시켰습니다.

각각 while 루프로 copy, paste를 실행해 race condition을 발생시킵니다.



그림 4. 동기화 탐지(copy)

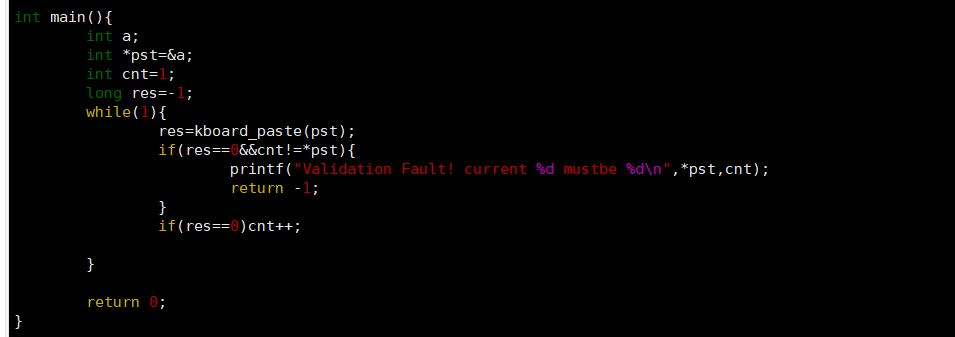


그림 5.동기화 탐지(paste)

race condition을 발생시키는 방법으로,

copy에서는 1부터 차례대로 값을 넣어주고,

paste로 한 개씩 값을 꺼내, 그 값이 차례로 1씩 증가하는지 검증하였습니다.

그 결과 기대한 값과 실제 paste한 값이 어긋나는 부분이 항상 존재했습니다.



그림 6. 동기화 문제 탐지

2-3. 동기화 문제 해결 및 수행결과

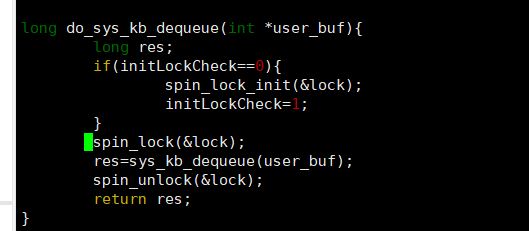


그림 7. 동기화문제 해결 주요 소스코드

동기화문제를 해결함에 있어, 과제안내에서 제공한 linux/spinlock.h 헤더를 사용했습니다.

이 떄, 위 사진처럼 spinlock을 초기화 해주어야 하는데, 1회만 수행되어야 하므로, initLockCheck 변수를 사용해서 lockCheck가 되었는지 확인해주었습니다.

그 후, Critical Section을 sys\_kb\_dequeue 함수 전체로 지정해, 위, 아래에 spin lock, unlock을 걸어주어, 임계구역을 수정할 때는, 한 프로세스만 접근할 수 있도록 하였습니다.

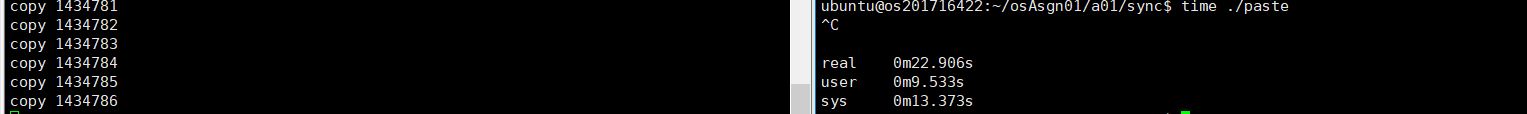


그림 8. 동기화 문제 해결

위 그림처럼, 오랜 시간동안, copy와 paste가 동시에 수행되어도 validation fault 없이 오랫동안 수행됨을 확인할 수 있었습니다.

2-4. 어려웠던 점

1. race condition을 탐지하는 방법에 대해, 어떤 방법을 사용할 지 고민이 많았으나, 가장 간단하게 쉘을 나누어 프로세스를 분할하는 방법을 택했습니다.
2. 동기화 문제를 해결할 때, init\_lock을 수행하는 위치에 대해 고민이 있었습니다.