

과제 #3 : Keyboard Interrupt

○ 과제 목표

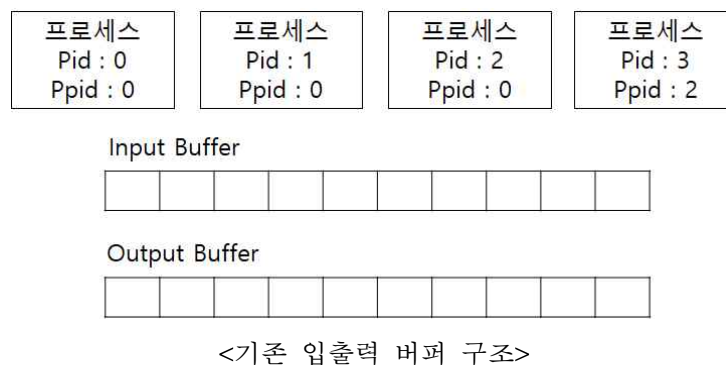
- Keyboard Interrupt의 수행 과정 이해 및 기능 추가
- 콘솔 클리어 기능 구현
- Foreground 프로세스 전환 구현

○ 기본 배경 지식

- Keyboard Interrupt
 - ✓ Keyboard Interrupt는 키보드 입력 시 발생하는 H/W Interrupt로 Keyboard Handler를 등록하여 제어할 수 있음
 - ✓ Handler 등록 시 인터럽트가 발생할 때마다 Handler가 자동으로 호출되며 내부에서 0x60포트를 읽어 입력된 키값을 가져올 수 있음
- 스캔코드
 - ✓ Keyboard Interrupt 발생시 0x60포트를 읽어 얻을 수 있는 값
 - ✓ Scancode는 2byte의 크기를 가지므로 ASCII로 변환하는 경우 주의 필요
- foreground process
 - ✓ 사용자와 상호작용을 하며 실행됨
 - ✓ 한 커맨드가 실행되는 동안 다른 커맨드는 실행이 안됨
- background process
 - ✓ 사용자와 상호작용을 하지 않고 실행
 - ✓ 키보드에 연결되지 않은 상태로 실행
 - ✓ 한 프로세스가 동작하는 동안 다른 프로세스가 실행될 수 있음

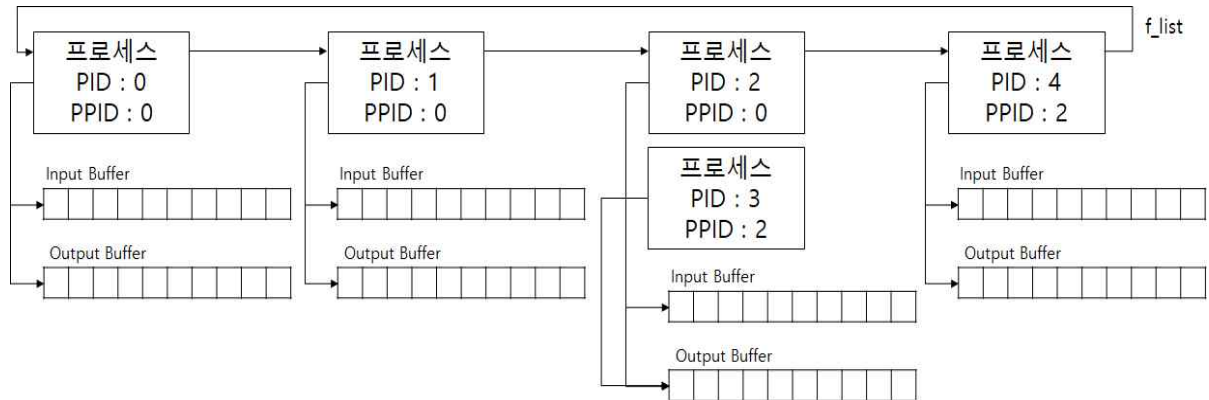
○ 과제 내용

- 기존 입출력



- ✓ 모든 프로세스가 동일한 입출력 버퍼를 사용
- ✓ 여러 프로세스가 존재해도 현재 프로세스가 종료되지 않을 경우 다른 프로세스 사용 불가

- 과제 후 입출력



<과제 수행 후 입출력 버퍼 구조>

- ✓ foreground 프로세스마다 입출력 버퍼를 할당
- ✓ 현재 foreground 프로세스의 출력 버퍼를 콘솔에 출력
- ✓ f_list로 foreground 프로세스를 관리
- ✓ 키보드 입력 및 콘솔 출력은 현재 foreground 프로세스에서 진행됨
 - cur_foreground_process 변수는 현재 foreground 프로세스를 가리킴
 - 출력 함수를 사용할 경우 현재 foreground 프로세스의 출력 버퍼가 아니라 해당 함수를 호출한 프로세스의 출력 변수를 사용함
 - 프로세스가 background 상태일 때 출력을 할 경우 콘솔에 출력되지 않다가 foreground 상태로 전환 시 콘솔에 출력이 됨
- ✓ 프로세스 생성 시 foreground 프로세스인지 background 프로세스인지 구분 관리
 - 프로세스 생성 시 옵션으로 해당 프로세스의 종류를 결정
 - foreground 프로세스인 경우
 - ☞ 입출력 버퍼를 할당받고 현재 foreground 프로세스가 됨
 - ☞ 프로세스 생성 시 foreground로 지정된 경우 f_list에 추가되고 관리됨
 - ☞ list 자료구조를 위해 struct process 구조체에 관련 변수가 추가됨
 - background일 경우 입력 버퍼는 할당받지 않고 출력 버퍼는 부모의 출력 버퍼를 사용함
 - ☞ pid가 3인 프로세스는 background 프로세스로 부모 프로세스의 출력 버퍼를 사용

○ 과제 수행 내용

- (1) 조합키 입력
 - ✓ 스캔코드는 키보드의 특정키를 누를 때와 떼를 때 각각 발생
 - ✓ 이를 이용하여 어떤 키가 눌렸는지 알 수 있음
 - ✓ Ctrl+l, Ctrl+Tab 조합키 구현
- (2) 콘솔 클리어
 - ✓ Ctrl+l 입력 시 콘솔 클리어 구현
 - ✓ 커서가 있는 라인이 콘솔 상에서 가장 위의 라인이 되도록 스크롤 구현
 - ✓ 출력 버퍼의 위치 값을 변경하여 콘솔 클리어 구현해야 함, 출력 함수를 통해 콘솔 클리어 구현할 경우 미구현으로 인정

- (3) foreground 프로세스 리스트 구현

- ✓ 프로세스 생성 시 foreground로 지정된 프로세스를 저장할 list는 기 구현된 f_list 변수를 사용
- 주어진 코드의 struct process 구조체에 f_list에서 사용할 멤버 변수(elem_foreground)가 선언되어 있음
- ✓ 생성된 프로세스가 foreground 프로세스일 경우 f_list에 추가
- ✓ Ctrl+Tab 입력 시 전역변수로 선언된 cur_foreground_proc가 다음 프로세스를 가리키도록 구현

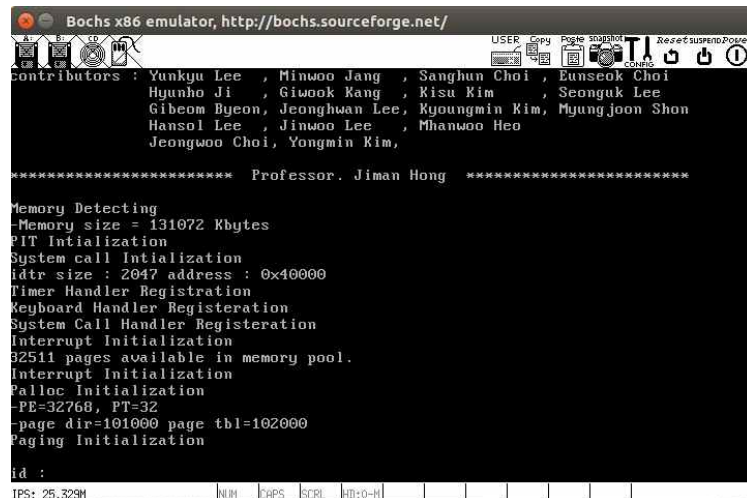
- (4) 입력 버퍼 전환

- ✓ 주어진 코드의 경우 모든 프로세스가 하나의 입력 버퍼(전역 변수)를 사용하고 있으나 본 과제에서는 foreground 프로세스를 생성할 때 마다 입력 버퍼를 할당할 수 있도록, 주어진 코드에서 선언된 struct Kbd_buffer 구조체를 사용하여 구현
- ✓ 프로세스 생성 시 옵션(foreground/background)에 따라 입력 버퍼 할당 방법이 달라야 함
- 주어진 코드의 struct process 구조체에 입력 버퍼를 할당받을 멤버 변수(kbd_buffer)가 선언되어 있음
- ✓ 주어진 코드를 struct Kbd_buffer 구조체를 사용하여 현재 foreground 프로세스의 입력 버퍼를 사용하는 방식으로 구현
- ✓ 2개의 프로세스 A와 B가 있을 때 콘솔이 A의 출력 버퍼만 보여주고 있을 경우 아래와 같이 동작해야 함
 - 현재 foreground 프로세스가 A이고 키보드로 임의의 입력을 할 경우 콘솔에 입력 값이 보임
 - Ctrl+Tab으로 현재 background 프로세스인 B를 foreground 프로세스로 변경하고 키보드로 임의의 입력을 할 경우 콘솔에 입력 값이 보이지 않음
 - Ctrl+Tab으로 현재 background 프로세스인 A를 foreground 프로세스로 변경하고 키보드로 임의의 입력을 할 경우 콘솔에 입력 값이 보임

- (5) 출력 버퍼 전환

- ✓ 주어진 코드의 경우 모든 프로세스가 하나의 출력 버퍼(buf_s)를 사용
- ✓ foreground 프로세스마다 출력 버퍼를 할당할 수 있도록 struct Console 구조체를 정의함
- ✓ 프로세스 생성 시 옵션에 따라 출력 버퍼 할당
 - struct process 구조체에 출력 버퍼를 할당받을 멤버 변수가 있음
 - foreground 프로세스가 아닌 경우 부모 프로세스의 출력 버퍼를 사용
- ✓ 주어진 코드를 struct Console 구조체를 사용하여 프로세스마다 지정된 출력 버퍼를 이용하도록 구현
- ✓ 입력 버퍼 전환 모듈을 구현한 상태에서 2개의 프로세스 A와 B가 있을 때 콘솔이 A의 출력 버퍼만 보여주고 있을 경우 아래와 같이 동작해야 함
 - 현재 foreground 프로세스가 A이고 키보드로 임의의 입력을 할 경우 콘솔에 입력 값이 보임
 - Ctrl+Tab으로 현재 background 프로세스인 B를 foreground 프로세스로 변경 할 경우 B의 출력 버퍼가 콘솔에 출력됨
 - 현재 foreground 프로세스인 B가 입력을 할 경우 입력 값이 콘솔에 출력됨
 - 현재 background 프로세스인 A가 출력을 할 경우 콘솔에 출력이 되지 않음
 - Ctrl+Tab으로 현재 background 프로세스인 A를 foreground 프로세스로 변경할 경우 콘솔에 출력 값이 출력됨

○ 과제 수행 후 실행 결과



Bochs x86 emulator, http://bochs.sourceforge.net/

Contributors : Yunkyu Lee , Minwoo Jang , Sanghun Choi , Eunseok Choi
 Hyunho Ji , Giwook Kang , Kisu Kim , Seonguk Lee
 Gibeom Byeon, Jeonghwan Lee, Kyoungmin Kim, Myungjoon Shon
 Hansol Lee , Jinwoo Lee , Mhanwoo Heo
 Jeongwoo Choi, Yongmin Kim,

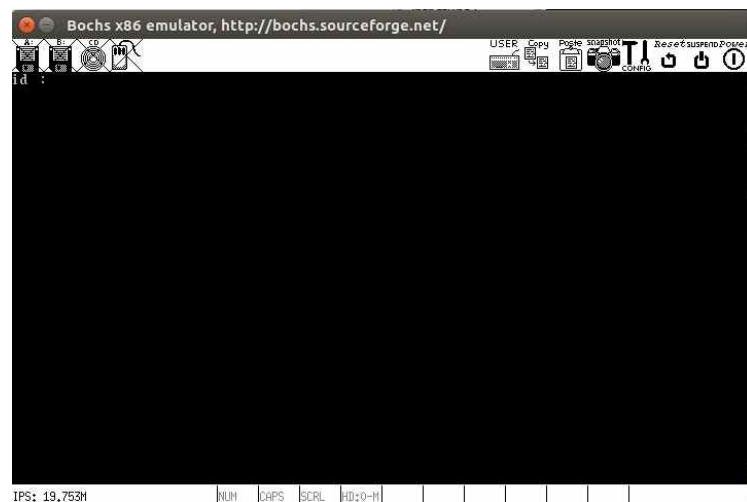
***** Professor. Jiman Hong *****

Memory Detecting
 -Memory size = 131072 Kbytes
 PIT Initialization
 System call Initialization
 Idtr size : 2047 address : 0x40000
 Timer Handler Registration
 Keyboard Handler Registration
 System Call Handler Registration
 Interrupt Initialization
 32511 pages available in memory pool.
 Interrupt Initialization
 Palloc Initialization
 -PE=32768, PT=32
 -page dir=101000 page tbl=102000
 Paging Initialization

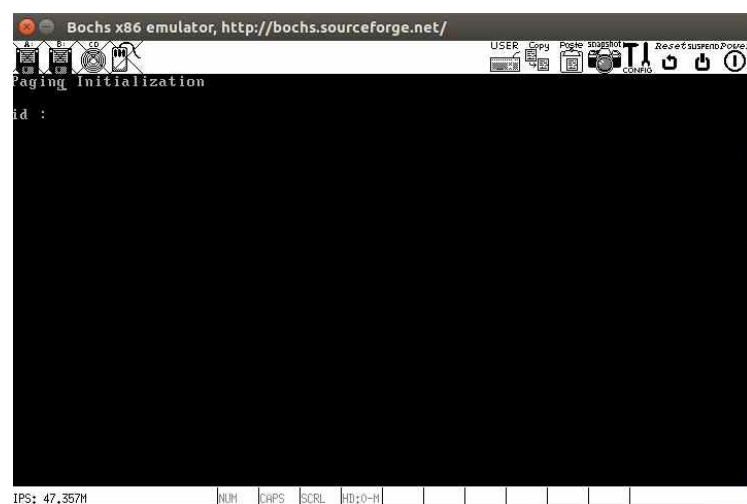
id :

IPS: 25,323M

<실행 후 초기 화면>



<Ctrl+I 입력 시 화면 클리어>



<화면 클리어 후 Page up 입력>

```
Bochs x86 emulator, http://bochs.sourceforge.net/

Memory Detecting
-Memory size = 131072 Kbytes
PIT Initialization
System call Initialization
idtr size : 2047 address : 0x40000
Timer Handler Registration
Keyboard Handler Registration
System Call Handler Registration
Interrupt Initialization
32511 pages available in memory pool.
Interrupt Initialization
Palloc Initialization
-PE=32768, PT=32
-page dir=101000 page tbl=102000
Paging Initialization

id : ssuos
password : oslab
> ps
pid 0 ppid non state 1 prio 0 using time 57 sched time 0
pid 1 ppid 0 state 1 prio 100 using time 9960 sched time 0
pid 2 ppid 1 state 1 prio 100 using time 151 sched time 0
> create_shell

IPS: 34.774M
```

<create_shell 명령어 새로운 프로세스 생성>

```
Bochs x86 emulator, http://bochs.sourceforge.net/

id :
```

<login shell이 생성되고 현재 foreground 프로세스가 변경>

```
Bochs x86 emulator, http://bochs.sourceforge.net/

Memory Detecting
-Memory size = 131072 Kbytes
PIT Initialization
System call Initialization
idtr size : 2047 address : 0x40000
Timer Handler Registration
Keyboard Handler Registration
System Call Handler Registration
Interrupt Initialization
32511 pages available in memory pool.
Interrupt Initialization
Palloc Initialization
-PE=32768, PT=32
-page dir=101000 page tbl=102000
Paging Initialization

id : ssuos
password : oslab
> ps
pid 0 ppid non state 1 prio 0 using time 57 sched time 0
pid 1 ppid 0 state 1 prio 100 using time 9960 sched time 0
pid 2 ppid 1 state 1 prio 100 using time 151 sched time 0
> create_shell
>
```

<Ctrl+tab 입력 시 콘솔 변경>

○ 과제제출 마감

- 2018년 9월 30일 (일) 23시 59분까지 제출

○ 배점 기준

- 보고서 15%
 - ✓ 개요 2%
 - ✓ 상세 설계 명세(기능 명세 포함) 10%
 - ✓ 실행 결과 3%
- 소스코드 85%
 - ✓ 컴파일 여부 5%(설계 요구에 따르지 않고 설계된 경우 0점 부여)
 - ✓ 실행 여부 80%(콘솔 클리어 15% + 입출력 버퍼 전환 65%)

○ 최소 구현사항

- 과제 수행 내용 (1), (2), (3), (4)