Lab. 8

Signal

Duksu Kim

공통사항

- **모든 System call 및 Standard C function 사용 가능**
  - 단, 외부 라이브러리 사용 불가
- **Source code 및 실행파일 이름은 문제 번호 사용**
  - 예) **8_1.c** 및 **7_1.out** 등
- **시간 측정 필요 시, 자신이 만든 측정 함수 사용**

- **EL 제출 시,**
  - 모든 source code를 모은 한글/워드 파일 별도 제출

Class 01

Type A

7 조

노유찬 김쾌남 이윤수

Lab 8-1. Signal Handler

- **문제 설명**

목적) Signal Handler의 사용법을 익히고 Signal의 종류를 확인해 본다.

- **1. Interrupt가 발생하게 되면**
- **2. x 초 후에 Alarm이 발생하게 된다.(x는 임의의 시간)**
- **3. Alarm clock 에서는 User defined signal 1, User defined signal 2가 발생한다.**
- **4. User defined signal 1에서는 signal에 해당하는 signal 설명 문자열을 1~99 까지 출력한다.**
- **5. User defined signal 2에서는 7_1.out이 실행하는 프로세스를 종료시킨다.**

Lab 8-1. Signal Handler

- **Hints**

- **SIGINT = Interrupt**
- **SIGALRM = Alarm clock**
- **SIGUSR1 = User defined signal 1**
- **SIGUSR2 = User defined signal 2**

Lab 8-1. Signal Handler

- 실행의 예

```
$ ./7_1.out
^C
Caught SIGINT!                                  // ctrl + c 발생
Received signal : Interrupt
(x 초 후…)
Caught SIGALRM!                                 // Alarm 시그널 발생
Received signal : Alarm clock
Caught SIGUSR1!                                 // SIGUSR1 시그널 발생
Received signal : User defined signal 1
Number #1 =  Hangup
Number #2 = Interrupt
…
Number #99 = Unknown signal 99
Caught SIGUSR2!                                 // SIGUSR2 시그널 발생
Received signal : User defined signal 2
Killed                                          // 프로세스 종료
```

Lab 8-2. Kill Process Game!

- **문제 설명**

  무한 루프를 도는 자식 프로세스를 생성하여
  끝나지 않는 자식 프로세스를 죽이는 게임!

- **필요한 지식**

  1. 기존 소스코드 해석 능력
  2. fork로 자식 프로세스 생성 후 execl() 함수 등으로 파일 실행하는 방법
  3. SIGKILL 로 프로세스를 없애는 방법

Lab 8-2. Kill Process Game!

- **구현해야 할 것**
  1. main 함수 내부
  2. kill_zombie() 메소드 – game.h 내부에 있음

```c
#include "game.h"

int main(void) {


        return 0;
}
```

```c
int kill_zombie() {


}
```

Lab 8-2. Kill Process Game!

- **Hints**
  1. game.h에 있는 함수들을 사용하시면 쉽습니다!
  2. 무한 루프 파일은 간단하게 만드시면 됩니다 - while(1)
  3. 좀비는 무한 루프를 도는 프로세스를 의미합니다.
  4. 좀비를 죽일 때는 같은 선상에 있을 때 죽습니다.
     → zomX = userX
  5. 실시간으로 움직이는 오브젝트를 보려면 while(1) 안에서 적절한 위치에 print_map()을 호출하세요.
  6. 기존 터미널 내용을 지우려면 terminal_clear()을 사용하세요.
  7. getch()는 game.h 내에 구현해놨습니다.

Lab 8-2. Kill Process Game!

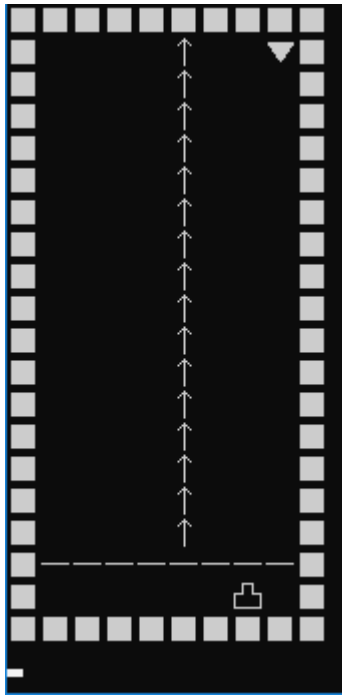
- **Hints**
  8. 입력을 원하시는 키로 바꾸시려면 input의
     아스키코드를 수정하시면 됩니다.
  9. game.h는 자유롭게 수정하셔도 좋습니다.
  10. 게임의 속도 조절은 sleep을 사용하세요.
  11. 게임 실행 시 기본 키는 이동 : a, d / 발사 : w 입니다.
  12. 맵의 크기는 자유롭게 입력하세요.

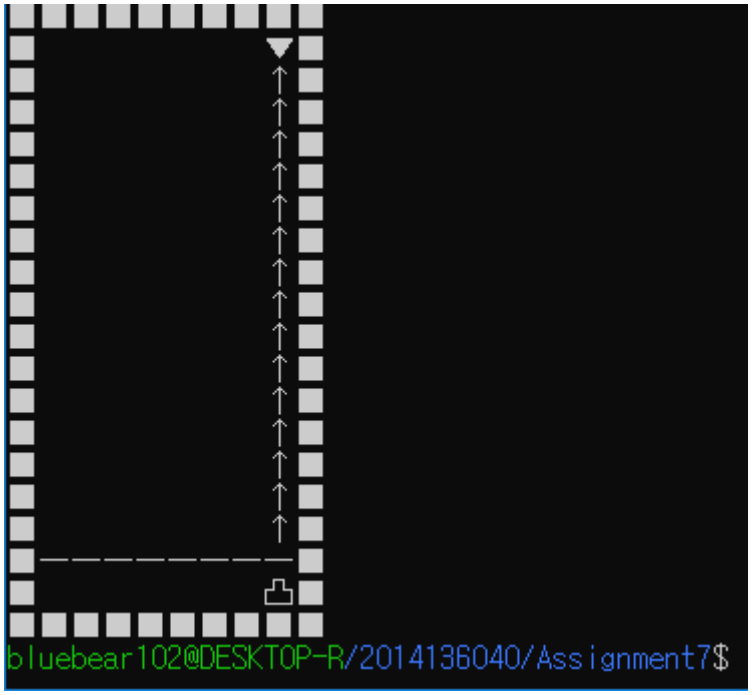
Lab 8-2. Kill Process Game!

- 실행의 예

Lab 8-2. Kill Process Game!

- 실행의 예





Lab 8-2. Kill Process Game!

- **제출 파일**
  - 8_2.c
  - game.h
  - while.c → 단순한 무한 루프 파일

- **주의 사항**
  - 실시간 게임이 아니라는 점
  - 제작기간 1일로 퀄리티가 매우 낮을 수 밖에 없다는 점
  - 또한, 퀄리티가 너무 낮다….^^7

Class 02

Type B

2조

노승현 나정기 김성호 원주혜

Lab 8-1. Signal Timer

- **문제 설명**
  - **시그널을 이용해 타이머 기능 구현**
    - 프로그램 실행 시 숫자를 받아서 시간을 설정한다.
    - 설정한 시간이 지나면 알람을 울린다.
      (5초간 울림 등 시간은 개인 재량)
    - 타이머 메뉴는 사용자 입력(CTRL+C)을 받아서 연다.
    - 타이머 메뉴에는 알람 미루기, 시간 재설정, 종료 기능이 존재한다.
    - 알람이 울리는 동안 사용자가 메뉴를 열지 않으면 자동으로 알람을 미룬다.

Lab 8-1. Signal Alarm

- **조건**
  - 타이머 기능을 구현할 때 alarm 함수 사용하기

Lab 8-1. Signal Alarm

- 실행 결과

```
wjh@wjh-VirtualBox:~/2015136081/Lab8$ ./8_2_1.out 5
Timer Start...
beep!
beep!
beep!
beep!
beep!
Execute auto-snooze
Timer Start...
beep!
beep!
^C
Input the menu number [1. Snooze 2. Set Time 3. End]: 2
Input the time(sec): 3
Timer Start...
beep!
beep!
beep!
^C
Input the menu number [1. Snooze 2. Set Time 3. End]: 1
Timer Start...
beep!
beep!
beep!
^C
Input the menu number [1. Snooze 2. Set Time 3. End]: 3
Timer Off
```

- 알람 울리는 방식은 5초동안 1초에 한 번씩 printf()로 문자 출력
- 5초 동안 CTRL+C를 입력하지 않으면 자동으로 다시 타이머 설정됨
- CTRL+C로 메뉴를 열어서
  1. 기존 설정 시간만큼 다시 타이머 설정
  2. 시간을 다시 설정
  3. 타이머 종료
- 기능 중 선택 가능하게 함

Lab 8-2. 등차수열 합 구하기

- **문제 설명**
  - a = 0, d = 1, n은 사용자 입력을 받고 등차수열의 합을 구하기
  - $1 \leq n \leq 256$

$$a_n = n-1 \text{ (단, } n\text{은 정수, } n > 0\text{)}$$

$$S_n = \sum_{k=1}^{n} a_k = 0+1+2+\cdots+a_n = \frac{n(2a+(n-1)d)}{2} = \frac{n(n-1)}{2}$$

- **Hints**
  - fork()
  - exit()

Lab 8-2. 등차수열 합 구하기

- **실행의 예**
  - 프로그램 종료 후 바로 다시 실행시켰을 때, Parent Process의 PID가 이전 프로그램의 마지막 Child Process의 PID의 다음 순서이어야함.

```
system@DESKTOP-MJ198RB:~/2013136033/Assignment07/Lab8_2$ ./8_2.out 5
You typed number : 5
Child Process - My PID : 144, My Parent's PID : 143
Child Process - My PID : 145, My Parent's PID : 143
Child Process - My PID : 146, My Parent's PID : 143
Child Process - My PID : 147, My Parent's PID : 143
Child Process - My PID : 148, My Parent's PID : 143
[Arithmetic Series]
a = 0, d = 1, n = 5, Result = 10
system@DESKTOP-MJ198RB:~/2013136033/Assignment07/Lab8_2$ ./8_2.out 10
You typed number : 10
Child Process - My PID : 150, My Parent's PID : 149
Child Process - My PID : 151, My Parent's PID : 149
Child Process - My PID : 152, My Parent's PID : 149
Child Process - My PID : 153, My Parent's PID : 149
Child Process - My PID : 154, My Parent's PID : 149
Child Process - My PID : 155, My Parent's PID : 149
Child Process - My PID : 156, My Parent's PID : 149
Child Process - My PID : 157, My Parent's PID : 149
Child Process - My PID : 158, My Parent's PID : 149
Child Process - My PID : 159, My Parent's PID : 149
[Arithmetic Series]
a = 0, d = 1, n = 10, Result = 45
```

Common Problem

Lab 8-3

By DS Kim

Lab 8-3. Print Prime Numbers

- **주어진 범위 안에, 소수를 모두 출력하는 병렬처리 프로그램**
  - 입력 : 수의 범위, 사용할 process 수, 출력 파일 명
- **입력한 수 만큼 process를 만들어서 병렬처리**
- **출력**
  - 범위 내의 소수가 **순서대로** 출력된 파일
- **Hints**
  - fork(), wait()

Lab 8-3. Print Prime Numbers

- 실행의 예

```
$ ./Lab8_3.out 2 100000 1 serial.txt
[P0] found 9592 primes
[Serial] takes 590.41 ms
$ ./Lab8_3.out 2 100 4 primeNumbers.txt
[P3] found 2762 primes
[P2] found 2371 primes
[P1] found 2260 primes
[P0] found 2199 primes
[Parallel] takes 252.34 ms
$ cat primeNumbers.txt
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
...
```

Extra Problem

Lab 8-X

By DS Kim

Kill All!

- **현재 사용자의 모든 Process를 종료 시키는 프로그램 작성**
  - 시스템 상에 자신이 만든 모든 process 종료
  - 이후, 위장용 source code에 대한 vi 창 자동으로 열기
    - 위장용 source code는 본인이 작성
  - Hints : /proc/#/status , readdir, Uid, ...