운영체제론 실습 6주차

CPS LAB

쓰레드를 이용한 Sudoku 답 검증기

Cyber Physical Systems Laboratory

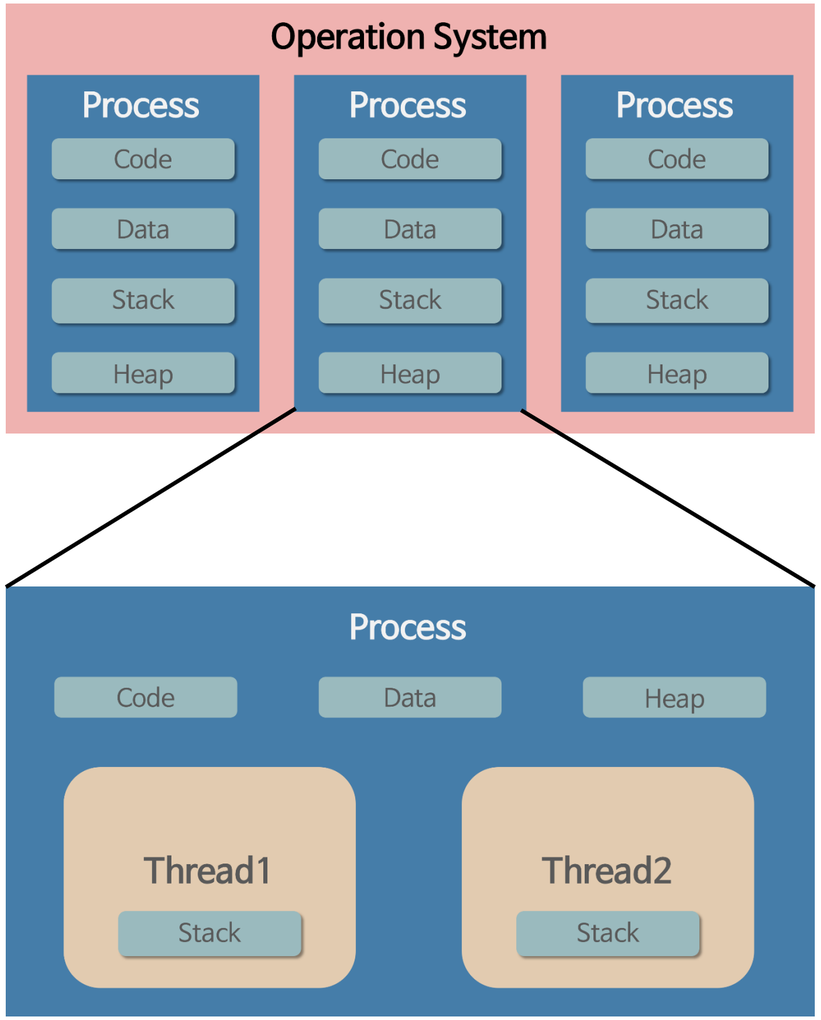
CHAPTER 1.

Thread

- Process vs Thread
- Thread 설명
- Thread 관련 함수 소개

Process vs Thread (1)

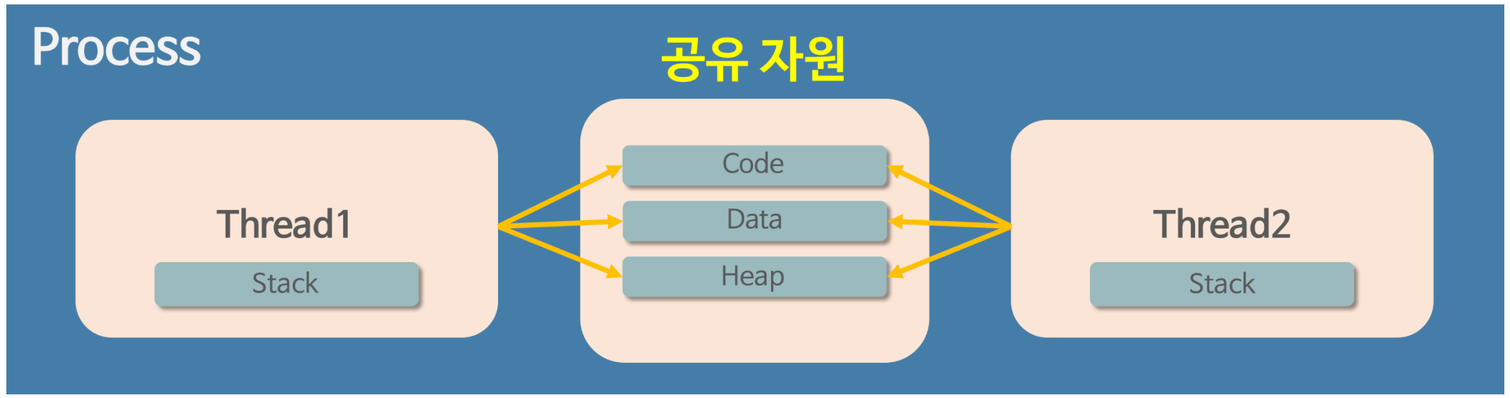
- **프로그램 (Program)**
  - 어떤 작업을 위해 실행할 수 있는 파일
  - **정적**인 상태 (**Static**)
- **프로세스 (Process)**
  - 운영체제로부터 자원을 할당 받은 **작업**의 단위
  - **동적**인 상태 (**Dynamic**)
- **쓰레드 (Thread)**
  - 프로세스가 할당 받은 자원을 이용하는 **실행 흐름**의 단위



Process vs Thread (2)

- **멀티 프로세스 (Multi-Process)**
  - 하나의 응용프로그램을 **여러 개의 프로세스**로 구성하여, 각 프로세스가 하나의 작업(task)을 처리하도록 하는 것

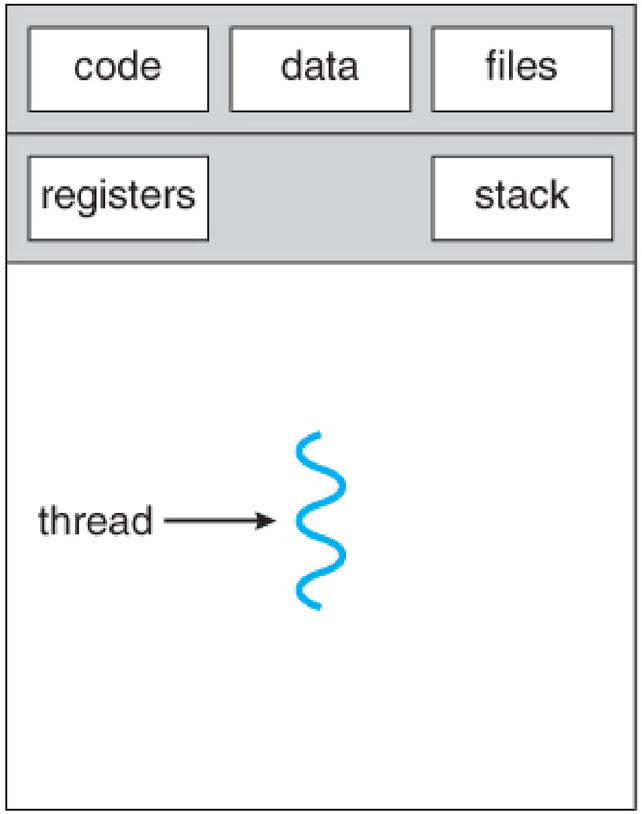
- **멀티 쓰레드 (Multi-Thread)**
  - 하나의 응용프로그램을 **여러 개의 쓰레드**로 구성하고, 각 쓰레드로 하여금 하나의 작업(task)을 처리하도록 하는 것



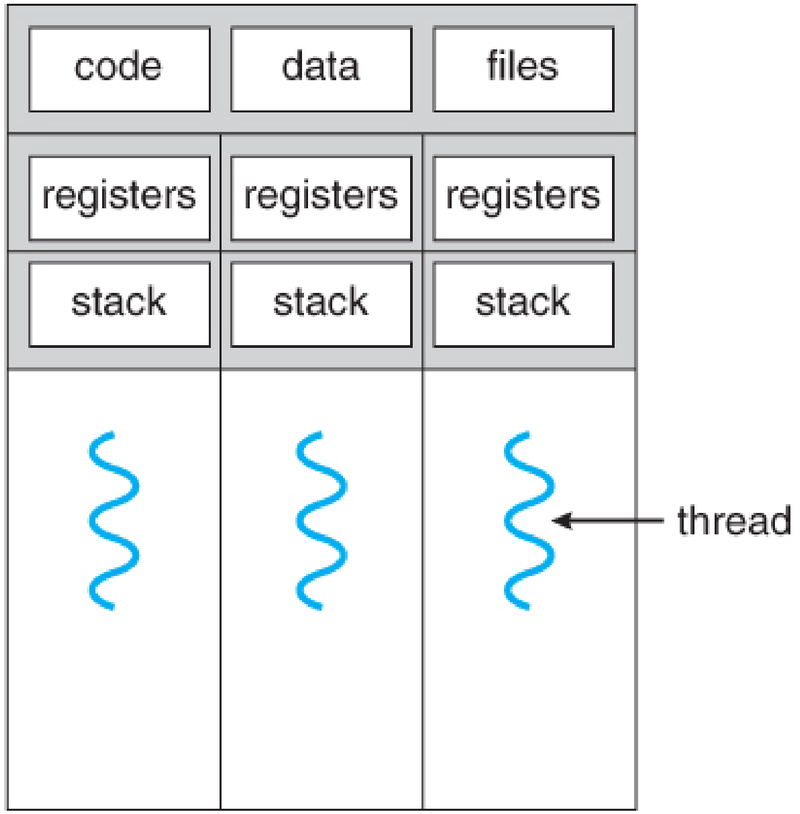
참고 : https://gmlwjd9405.github.io/2018/09/14/process-vs-thread.html

Thread

- 프로세스 내에서 실행되는 여러 흐름의 단위
- 프로세스 1개당 최소 1개의 쓰레드가 존재



single-threaded process



multithreaded process

pthread 함수

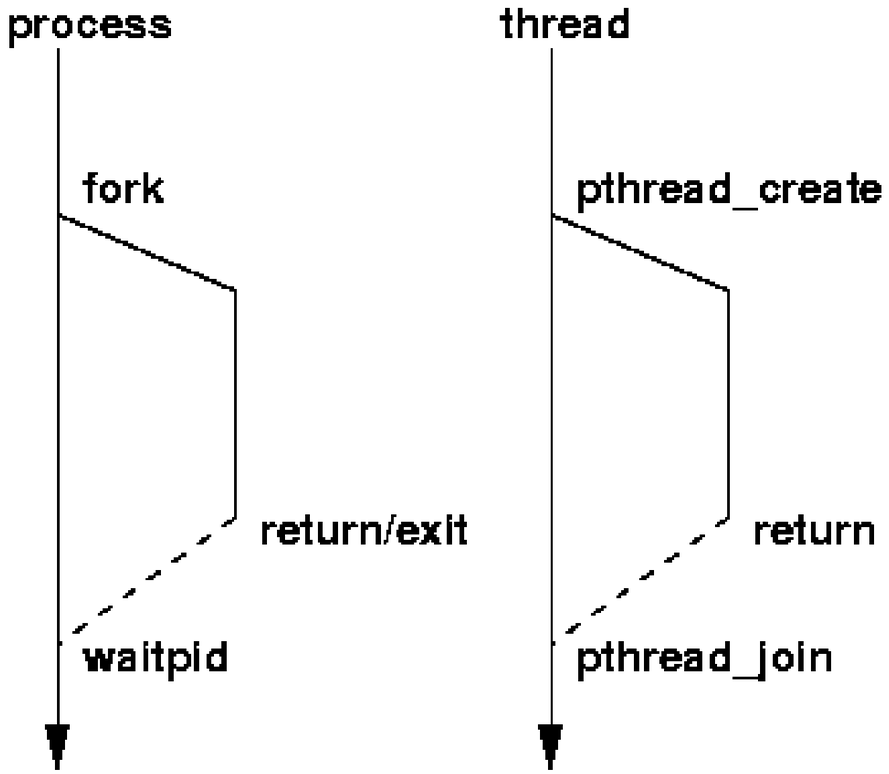
- POSIX thread
- POSIX에서 표준으로 제안한 thread 함수 모음

- **기본 pthread 함수**

  pthread_create() : thread를 생성한다.

  pthread_exit() : thread를 종료한다.

  pthread_join() : thread 종료를 기다린다.



pthread_create

- int **pthread_create**(pthread_t ***thread**, const pthread_attr_t ***attr**,
  void *(***start_routine**)(void *), void ***arg**);

➢ 새로운 쓰레드를 생성한다.

1) pthread_t *thread

- 쓰레드가 성공적으로 생성되었을 때 생성된 쓰레드를 식별하기 위해 사용되는 쓰레드 식별자

2) const pthread_attr_t *attr

- 쓰레드 특성을 지정, default로 지정할 경우 NULL로 설정

3) void *(*start_routine)(void *)

- 쓰레드가 실행할 함수

4) void *arg

- start_routine으로 넘어갈 매개변수

5) Return Value

- 0 : 정상종료 , 0이 아닌 값 : 에러

pthread_exit

- void **pthread_exit**(void ***retval**);

➢ 쓰레드를 종료한다.

1) void *retval

- 쓰레드 종료 시 자원을 반납해야 하는 등의 처리작업이 필요할 때,
  pthread_cleanup_push()를 사용해 콜백함수를 등록해 이 인자를 사용
- 사용하지 않아도 되는 경우 NULL

pthread_join

- int **pthread_join**(pthread_t **th**, void ****thread_return**);

➢ th 번호를 가지는 쓰레드의 종료를 기다린다.

1) pthread_t th

- th를 식별 번호로 가지는 쓰레드가 종료될 때까지 기다린다.

2) void **thread_return

- thread_return값이 NULL이 아닌 경우, th를 식별 번호로 가지는 쓰레드의 return값을 받아 옴

3) Return Value

- 성공 : 0 , 실패 : Error Code

Example Code (1)

week6/1_sum_thread/sum_thread.c

```c
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>

int sum;
void *runner(void *param);

int main(int argc, char *argv[])
{
        pthread_t tid;
        pthread_attr_t attr;

        // 공백 입력 or 다중 입력 예외 처리
        if (argc != 2) {
                fprintf(stderr, "usage: ./sum_pthread <integer value>\n");
                return -1;
        }

        // 음수 입력 예외 처리
        if (atoi(argv[1]) < 0) {
                fprintf(stderr, "input value : [%d] must be >= 0\n", atoi(argv[1]));
                return -1;
        }

        // 쓰레드 속성 객체 attr 초기화
        pthread_attr_init(&attr);

        /* create thread */
        // 쓰레드(tid)에서 argv[1]을 매개변수로 하여 runner 함수 실행
        pthread_create(&tid, &attr, runner, argv[1]);

        /* wait for all created threads */
        // 쓰레드(tid)가 종료될 때 까지 대기
        pthread_join(tid, NULL);

        printf("sum = %d\n", sum);
}
```

```c
void *runner(void *param)
{
        int i, upper = atoi(param);
        sum = 0;

        for (i = 1; i <= upper; i++)
                sum += i;

        /* exit thread */
        // 쓰레드 종료
        pthread_exit(0);
}
```

Example Code (2)

week6/1_sum_thread/sum_thread.c

- 컴파일 및 코드 실행

```
$ make
$ ./sum_thread {number}
```

```
os@os-virtual-machine:~/os-week/week6/1_sum_thread$ make
gcc -g -o sum_thread sum_thread.c -lpthread
os@os-virtual-machine:~/os-week/week6/1_sum_thread$ ./sum_thread 10
sum = 55
```

➢ Result : 1~{number} 까지의 합 출력

CHAPTER 2.

Project

- Sudoku 답 검증기 프로젝트 설명
- Skeleton Code
- 결과 화면

Sudoku 검증기 설명 (1)

- **2차원 배열로 만들어진 스도쿠의 유효성을 판단**
- **아래 조건별로 숫자 1~9를 갖는지 쓰레드를 생성해 검사**

1. 3x3 Matrix
2. Column
3. Row

```
int sudoku[9][9] = {
    {6, 2, 4, 5, 3, 9, 1, 8, 7},
    {5, 1, 9, 7, 2, 8, 6, 3, 4},
    {8, 3, 7, 6, 1, 4, 2, 9, 5},
    {1, 4, 3, 8, 6, 5, 7, 2, 9},
    {9, 5, 8, 2, 4, 7, 3, 6, 1},
    {7, 6, 2, 3, 9, 1, 4, 5, 8},
    {3, 7, 1, 9, 5, 6, 8, 4, 2},
    {4, 9, 6, 1, 8, 2, 5, 7, 3},
    {2, 8, 5, 4, 7, 3, 9, 1, 6}
};
```

Sudoku 검증기 설명 (2)

- 3x3 크기마다 1~9의 숫자를 모두 가지고 있는지 검사

※ 검증 기준 :

[0][0], [0][3], [0][6],

[3][0], [3][3], [3][6]

[6][0], [6][3], [6][6]

```
int sudoku[9][9] = {
[0][n] {6, 2, 4, 5, 3, 9, 1, 8, 7},
       {5, 1, 9, 7, 2, 8, 6, 3, 4},
       {8, 3, 7, 6, 1, 4, 2, 9, 5},
[3][n] {1, 4, 3, 8, 6, 5, 7, 2, 9},
       {9, 5, 8, 2, 4, 7, 3, 6, 1},
       {7, 6, 2, 3, 9, 1, 4, 5, 8},
[6][n] {3, 7, 1, 9, 5, 6, 8, 4, 2},
       {4, 9, 6, 1, 8, 2, 5, 7, 3},
       {2, 8, 5, 4, 7, 3, 9, 1, 6}
};   [n][0]      [n][3]      [n][6]
```

Sudoku 검증기 설명 (3)

- 세로(Column)가 1~9의 숫자를 모두 가지고 있는지 검사

※ 검증 기준 :

[0][0], [0][1], [0][2],

[0][3], [0][4], [0][5]

[0][6], [0][7], [0][8]

```
int sudoku[9][9] = {
[0][n] {6, 2, 4, 5, 3, 9, 1, 8, 7},
       {5, 1, 9, 7, 2, 8, 6, 3, 4},
       {8, 3, 7, 6, 1, 4, 2, 9, 5},
[3][n] {1, 4, 3, 8, 6, 5, 7, 2, 9},
       {9, 5, 8, 2, 4, 7, 3, 6, 1},
       {7, 6, 2, 3, 9, 1, 4, 5, 8},
[6][n] {3, 7, 1, 9, 5, 6, 8, 4, 2},
       {4, 9, 6, 1, 8, 2, 5, 7, 3},
       {2, 8, 5, 4, 7, 3, 9, 1, 6}
};   [n][0]      [n][3]      [n][6]
```

Sudoku 검증기 설명 (4)

- 가로(Row)가 1~9의 숫자를 모두 가지고 있는지 검사

※ 검증 기준 :

[0][0], [1][0], [2][0],

[3][0], [4][0], [5][0]

[6][0], [7][0], [8][0]

```
int sudoku[9][9] = {
[0][n] {6, 2, 4, 5, 3, 9, 1, 8, 7},
       {5, 1, 9, 7, 2, 8, 6, 3, 4},
       {8, 3, 7, 6, 1, 4, 2, 9, 5},
[3][n] {1, 4, 3, 8, 6, 5, 7, 2, 9},
       {9, 5, 8, 2, 4, 7, 3, 6, 1},
       {7, 6, 2, 3, 9, 1, 4, 5, 8},
[6][n] {3, 7, 1, 9, 5, 6, 8, 4, 2},
       {4, 9, 6, 1, 8, 2, 5, 7, 3},
       {2, 8, 5, 4, 7, 3, 9, 1, 6}
};   [n][0]      [n][3]      [n][6]
```

Skeleton Code

week6/2_project/sudoku_validator.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>

#define num_threads 27

// 각 쓰레드의 valid check가 유효한지 저장하기 위한 int list
// 0이면 invalid, 1이면 valid
int valid[num_threads] = {0};

typedef struct {
        int row;
        int column;
} parameters;

// Sudoku puzzle to be solved
int sudoku[9][9] = {
        {6, 2, 4, 5, 3, 9, 1, 8, 7},
        {5, 1, 9, 7, 2, 8, 6, 3, 4},
        {8, 3, 7, 6, 1, 4, 2, 9, 5},
        {1, 4, 3, 8, 6, 5, 7, 2, 9},
        {9, 5, 8, 2, 4, 7, 3, 6, 1},
        {7, 6, 2, 3, 9, 1, 4, 5, 8},
        {3, 7, 1, 9, 5, 6, 8, 4, 2},
        {4, 9, 6, 1, 8, 2, 5, 7, 3},
        {1, 8, 5, 4, 7, 3, 9, 1, 6}
};

// Method that determines if numbers 1-9 only appear once in a column
void *isColumnValid(void* param) {
        // Confirm that parameters indicate a valid col subsection
        parameters *params = (parameters*)param;
        int row = params->row;
        int col = params->column;

        // 아래 두 경우에 스레드를 종료시킴.
```

- **Skeleton Code의 TODO 부분 채우기**

① TODO 1 ~ 9 : exit_thread

② TODO 10 ~ 12 : create thread

③ TODO 13 : wait for all created thread

Sudoku 검증기 결과 화면

```
os@os-virtual-machine:~/os-week/week6/sudoku_validator_ans$ ./sudoku_validator
 1th thread created with 'is3x3Valid' function at       [0][0]:6
 2th thread created with 'isColumnValid' function at    [0][0]:6
 3th thread created with 'isRowValid' function at       [0][0]:6
 4th thread created with 'isColumnValid' function at    [0][1]:2
 5th thread created with 'isColumnValid' function at    [0][2]:4
 6th thread created with 'is3x3Valid' function at       [0][3]:5
 7th thread created with 'isColumnValid' function at    [0][3]:5
 8th thread created with 'isColumnValid' function at    [0][4]:3
 9th thread created with 'isColumnValid' function at    [0][5]:9
10th thread created with 'is3x3Valid' function at       [0][6]:1
11th thread created with 'isColumnValid' function at    [0][6]:1
12th thread created with 'isColumnValid' function at    [0][7]:8
13th thread created with 'isColumnValid' function at    [0][8]:7
14th thread created with 'isRowValid' function at       [1][0]:5
15th thread created with 'isRowValid' function at       [2][0]:8
16th thread created with 'is3x3Valid' function at       [3][0]:1
17th thread created with 'isRowValid' function at       [3][0]:1
18th thread created with 'is3x3Valid' function at       [3][3]:8
19th thread created with 'is3x3Valid' function at       [3][6]:7
20th thread created with 'isRowValid' function at       [4][0]:9
21th thread created with 'isRowValid' function at       [5][0]:7
22th thread created with 'is3x3Valid' function at       [6][0]:3
23th thread created with 'isRowValid' function at       [6][0]:3
24th thread created with 'is3x3Valid' function at       [6][3]:9
25th thread created with 'is3x3Valid' function at       [6][6]:8
26th thread created with 'isRowValid' function at       [7][0]:4
27th thread created with 'isRowValid' function at       [8][0]:2
```

```
 0th thread terminated
 1th thread terminated
 2th thread terminated
 3th thread terminated
 4th thread terminated
 5th thread terminated
 6th thread terminated
 7th thread terminated
 8th thread terminated
 9th thread terminated
10th thread terminated
11th thread terminated
12th thread terminated
13th thread terminated
14th thread terminated
15th thread terminated
16th thread terminated
17th thread terminated
18th thread terminated
19th thread terminated
20th thread terminated
21th thread terminated
22th thread terminated
23th thread terminated
24th thread terminated
25th thread terminated
26th thread terminated
Sudoku solution is valid!
```

Sudoku가 올바른 답을 가지고 있으면 valid, 아니라면 invalid

감사합니다.

CPS LAB

Lim Jiseoup

jseoup@hanyang.ac.kr