INHA UNIVERSITY

Operating System

thread 과제

과 목 명 : opearting system
담당교수 : 송민석 교수님
제 출 일 : 2017년 04월 22일

인하대학교 공과대학
컴퓨터정보공학 3학년
12131597 정민교

목차

1.sudoku

1.1 프로그램 설명

input.txt로 받은 9x9 행렬이 sudoku인지 아닌지 판별하는 프로그램.
세로줄에 1부터 9까지 중복되는 수가 있는지 확인하는 thread 1개,
가로줄에 1부터 9까지 중복되는 수가 있는지 확인하는 thread 1개,
3x3 모양의 격자에 중복되는 수가 있는지 확인하는 thread 9개를
통하여 중복되는 수가 없을 경우 각 thread에서 0을 return,
그렇지 못하는 경우에는 1을 return해주어서 마지막 main thread에서
그 값들을 통해 sudoku가 valid하는 지 판별한다.

1.2 thread 함수

(1) checkrow 함수

```
void *checkrow(void *p)
{
    printf("check row thread start");
    int arg = (int) p;
    int check[10]={0};
    pthread_t tid;
    tid = pthread_self();
    int i,j;
    int k=0;
    for(i=0; i<9; i++)
    {
        for(j=0; j<9; j++)
        {
            check[sudoku[i][j]]++;
        }

        for(j=1; j<=9; j++)
        {
            if(check[j] != 1)
            {
                k++;
                break;
            }
        }

        for(j=1; j<=9; j++)
        {
            check[j]=0;
        }
    }
    pthread_exit((void*) k);
}
```

1. 첫 번째 가로 줄부터 input값을 확인한다.

2. input값에 해당하는 check배열의 index의 값을 1씩 올려준다.
3. check 배열의 값을 하나씩 확인하면서 배열의 값이 1이 아닌 경우 k값(초기값0)을 1을 올려준 다음 반복문을 빠져나간다.
4. 첫 번째 가로줄부터 시작해서 마지막 가로줄까지 1 – 3을 반복한다.
5. k 값을 thread를 끝내면서 return 해준다.

(2) checkcol 함수

```
void *checkcol(void *p)
{
	printf("check col thread start");
	int arg = (int) p;
	int check[10]={0};
	pthread_t tid;
	tid = pthread_self();
	int i,j;
	int k=0;
	for(i=0; i<9; i++)
	{
		for(j=0; j<9; j++)
		{
			check[sudoku[j][i]]++;
		}

		for(j=1; j<=9; j++)
		{
			if(check[j] != 1)
			{
				k++;
				break;
			}
		}

		for(j=1; j<=9; j++)
		{
			check[j]=0;
		}
	}
	pthread_exit((void*) k);
}
```

1. 첫 번째 세로 줄부터 input값을 확인한다.
2. input값에 해당하는 check배열의 index의 값을 1씩 올려준다.
3. check 배열의 값을 하나씩 확인하면서 배열의 값이 1이 아닌 경우 k값(초기값0)을 1을 올려준 다음 반복문을 빠져나간다.
4. 첫 번째 세로줄부터 시작해서 마지막 세로줄까지 1 – 3을 반복한다.
5. k 값을 thread를 끝내면서 return 해준다.

(3) checkgrid 함수

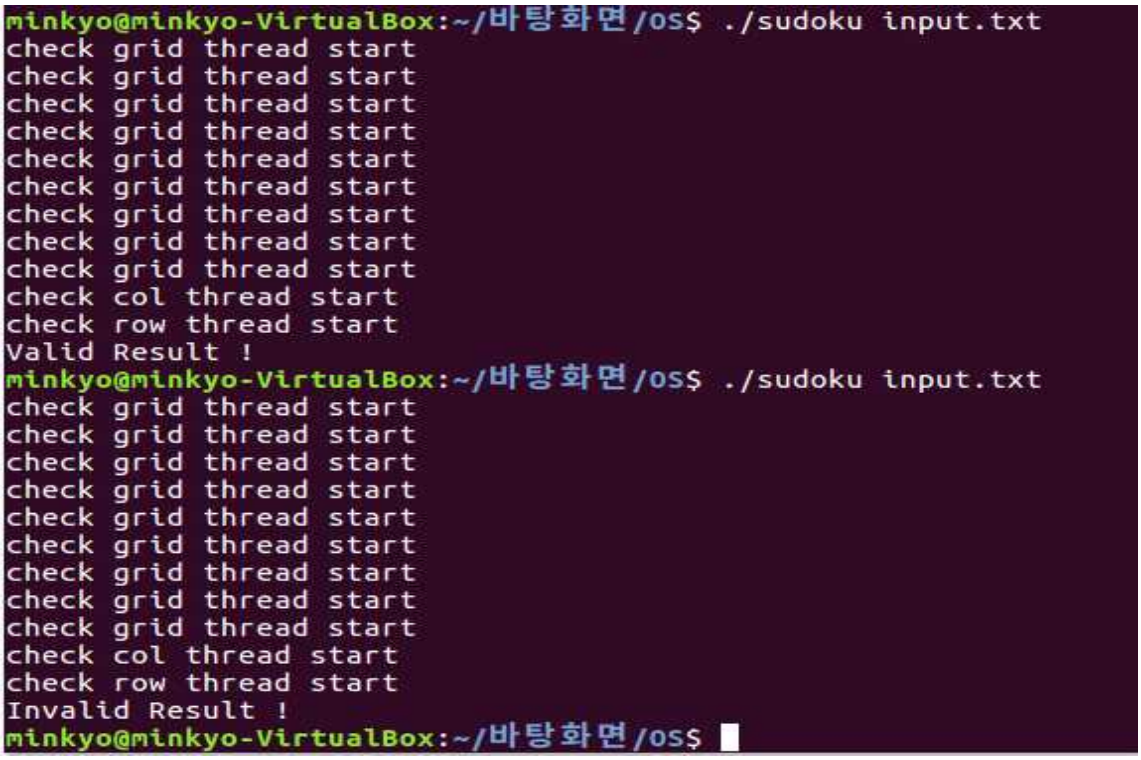
```
void *checkgrid(void *p)
{
    printf("check grid thread start");
    int check[10] = {0};
    pthread_t tid;
    tid = pthread_self();
    struct data *Dt;
    int row,col;
    int i,j;
    int k=0;
    Dt = (struct data*) p;
    row = Dt->row;
    col = Dt->col;

    for(i=row-2; i<=row; i++)
    {
        for(j=col-2; j<=col; j++)
        {
            check[sudoku[i][j]]++;
        }
    }

    for(j=1; j<=9; j++)
    {
        if(check[j] != 1)
        {
            k++;
            break;
        }
    }
    pthread_exit((void*) k);
}
```

1. 첫 번째 grid부터 input값을 확인한다.
2. input값에 해당하는 check배열의 index의 값을 1씩 올려준다.
3. check 배열의 값을 하나씩 확인하면서 배열의 값이 1이 아닌 경우 k값(초기값0)을 1을 올려준 다음 반복문을 빠져나간다.
4. k 값을 thread를 끝내면서 return 해준다.
5. 9개의 grid를 확인하기 위해 9개의 thread를 생성한 후 1 – 4를 반복한다.

1.3 결과



각 thread는 시작하면서 스레드가 동작하는지 여부를 printf문을 출력하면서 시작

각 thread 값을 main thread에서 return 받은 후 변수 correct에 저장

correct가 0일 경우 valid Result를 출력

correct가 0이 아닌 경우 invalid Result를 출력

2.fibonacci

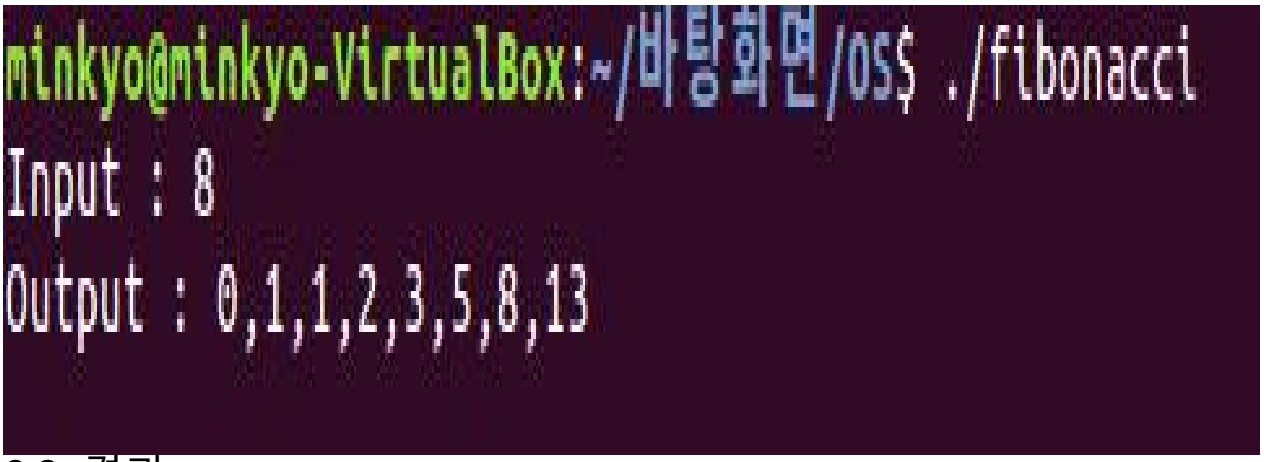
2.1 프로그램 설명

입력 값으로 받은 숫자만큼 fibonacci 수를 생성하여 출력하는 프로그램. main thread에서 input 입력값을 입력받은 후, thread를 생성하여 입력받은 수만큼 배열을 생성하여 fibonacci수를 생성한 후 결과를 출력한다.

2.2 thread 함수

```
void *fibo(void *num)
{
    pthread_t tid;
    tid = pthread_self();
    int *fibo_arr;
    int size = *(int *)num;
    int i;
    fibo_arr = (int*)malloc(sizeof(int)*size);
    fibo_arr[0] = 0;
    fibo_arr[1] = 1;
    for(i=2; i<size; i++)
    {
        fibo_arr[i] = fibo_arr[i-1] + fibo_arr[i-2];
    }
    printf("Output : ");
    for(i=0; i<size; i++)
    {
        if(i==size-1)
        printf("%d₩n",fibo_arr[i]);
        else
        printf("%d,",fibo_arr[i]);
    }
    printf("₩n");
}
```

1. main thread에서 입력 받은 input 값을 파라미터로 넘겨받아서 그 수만큼 배열을 동적할당한다.
2. 반복문을 돌면서 fibonacci 수를 생성한다.
3. output을 출력한다.


minkyo@minkyo-VirtualBox:~/바탕화면/os$ ./fibonacci
Input : 8
Output : 0,1,1,2,3,5,8,13

2.3 결과