




6주차 퀴즈

 Date	@2023년 1월 11일
 Tags	취준스터디
 출처	
 참고자료	

HTTP

HTTP 특징 중 상태유지와 무상태에 대해 설명하고 둘의 차이점에 대해 설명해주세요.

상태유지란 서버가 클라이언트의 상태를 항상 가지고 있어서 클라이언트와 통신하는데 있어 문맥을 기억하고 있는 상태를 말한다.

무상태의 경우 서버가 클라이언트의 상태를 가지고 있지 않고 클라이언트가 자신의 상태를 항상 말해주는 형태의 통신을 말한다.

둘의 차이점으로는 서버가 변경되는 경우를 들어 말할 수 있는데, 만약, 상태유지의 경우 서버가 변경되거나 다운되는 상황이 발생하면 클라이언트의 상태데이터를 가지고 있던 서버에서 데이터를 가져오거나 해야하기 때문에 서버가 변경되는 상황에 대응하기가 어렵다. 반면에, 무상태의 경우 서버가 클라이언트의 상태를 가지고 있지 않기 때문에 서버가 다운되던가 추가되는 상황에서 자유롭다. 단, 클라이언트에서 서버에 보내야 할 데이터가 비교적 많아지는 상황이 발생한다.

무상태의 한계와 무상태로 상태유지를 어떻게 구현하는지 설명해주세요

무상태로 설계하기 힘든 것에 대표적으로 로그인이다. 로그인의 경우 사용자가 로그인 했다는 상태를 서버에 유지해야한다 이러한 경우 브라우저 쿠키와 서버 세션을 사용하여 상태유지를 한다. 이런 경우에 한해서만 상태유지를 사용한다.

비연결성이 무엇인지 말하고, 비연결성의 한계점을 어떻게 극복했는지 설명하세요

비연결성의 경우 연결을 유지하지 않는 것을 말한다. http 의 경우 비연결성을 가지고 있는데 연결을 유지하지 않기 때문에 원하는 응답을 받으면 연결을 끊어버린다. 이러한 특징은 자원을 효율적으로 사용할 수 있다는 장점이 있는 반면에 받아야하는 자원이 너무 많을 때는 비효율적으로 작동한다. 이를테면 홈페이지에 너무 많은 데이터가 뿌려져있는 경우가 그에 해당한다. 그럴경우 요청 여러번을 처리하기 위해서 통신을 유지하고 있다가 모든 데이터 요청을 처리하면 연결을 끊는다!

구현

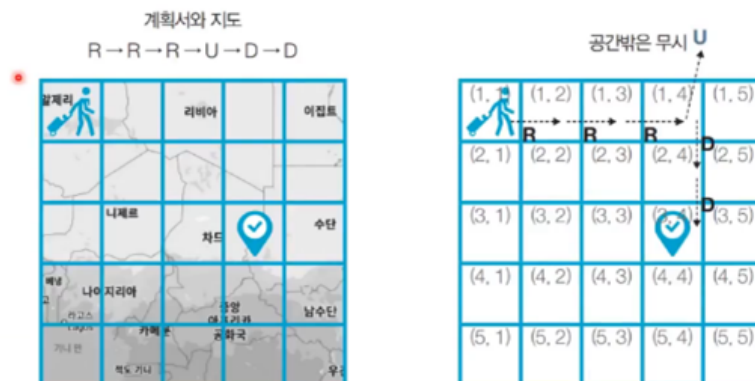
Quiz

***2개의 문제 중에서 본인이 풀 수 있는 문제를 골라서 풀이하세요 (둘 다 풀어도 상관없습니다)**

#1. 상하좌우 (난이도 ★ 출처. 이것이 코딩테스트다 파이썬편)

- 여행가 A는 $N \times N$ 크기의 정사각형 공간 위에 서 있습니다. 이 공간은 1×1 크기의 정사각형으로 나누어져 있습니다. 가장 왼쪽 위 좌표는 (1, 1)이며, 가장 오른쪽 아래 좌표는 (N, N)에 해당합니다. 여행가 A는 상, 하, 좌, 우 방향으로 이동할 수 있으며, 시작 좌표는 항상 (1, 1)입니다. 우리 앞에는 여행가 A가 이동할 계획이 적힌 계획서가 놓여 있습니다.
- 계획서에는 하나의 줄에 띄어쓰기를 기준으로 하여 L, R, U, D 중 하나의 문자가 반복적으로 적혀 있습니다. 각 문자의 의미는 다음과 같습니다.
 - L: 왼쪽으로 한 칸 이동
 - R: 오른쪽으로 한 칸 이동
 - U: 위로 한 칸 이동
 - D: 아래로 한 칸 이동

- 이때 여행가 A가 $N \times N$ 크기의 정사각형 공간을 벗어나는 움직임은 무시됩니다. 예를 들어 (1, 1)의 위치에서 L 혹은 U를 만나면 무시됩니다. 다음은 $N = 5$ 인 지도와 계획서입니다.



- 입력 조건**

 - 첫째 줄에 공간의 크기를 나타내는 N 이 주어집니다. ($1 \leq N \leq 100$)
 - 둘째 줄에 여행가 A가 이동할 계획서 내용이 주어집니다. ($1 \leq$ 이동 횟수 ≤ 100)

출력 조건

 - 첫째 줄에 여행가 A가 최종적으로 도착할 지점의 좌표 (X, Y)를 공백을 기준으로 구분하여 출력합니다.

입력 예시

5
R R R U D D

출력 예시

3 4

```

from sys import stdin

N = 5 #map(int,stdin.readline());

arr = ['R','R','R','U','D','D'] #list(map(int,stdin.readline()));

x = 1
y = 1

mapx = [0,0,-1,1]
mapy = [-1,1,0,0]

#왼쪽, 오른쪽, 위, 아래

memo = {'L':0, 'R':1, 'U':2, 'D':3}

for value in arr :
    idx = memo[value]
    if x + mapx[idx] == 0 or y+mapy[idx] == 0:
        continue
    else :
        x += mapx[idx]
        y += mapy[idx]

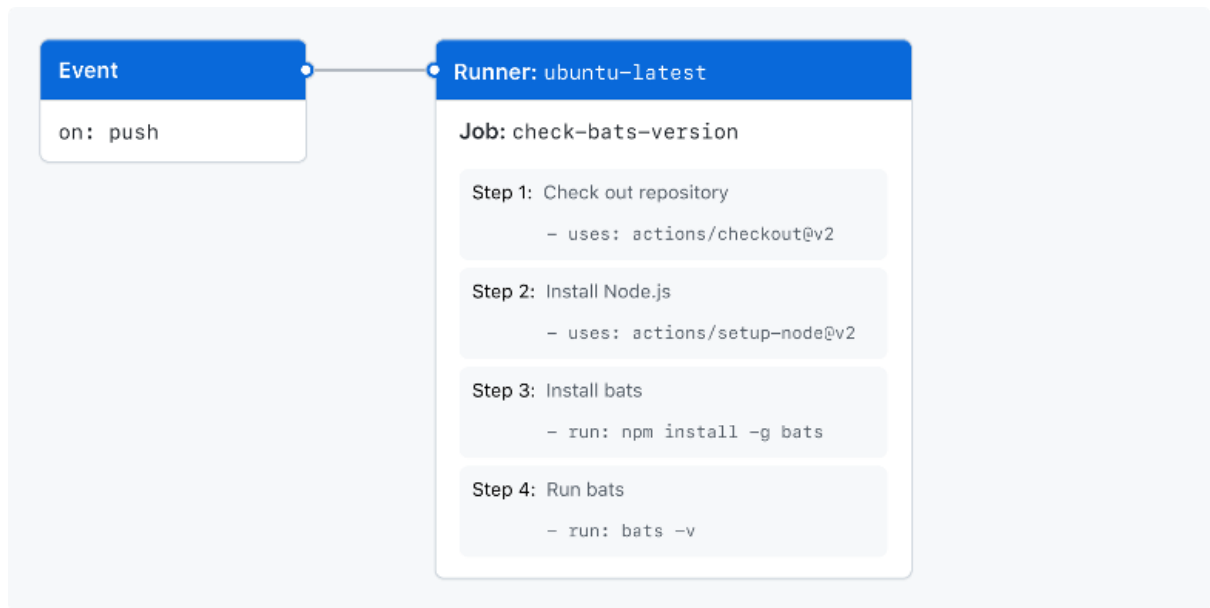
print(x,y)

//다른 범위를 넘어가는 조건을 추가하는게 좋겠음

```

Git hub action

github action workflow를 .yml으로 작성



```
name:

on:
  [push]

jobs:
  check-bats-version:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2
      - uses: actions/setup-node@v2

      - name: install nodejs
        run: npm install -g bats

      - name: Run bats
        run: bats -v
```

CAP 이론

참고자료

1. CAP 이론에 대해 간단히 설명하고, 오늘날 이 이론을 바탕으로 DB를 선정하는 데에 한계에 부딪히게 된 이유를 설명하세요.

CAP이론이란 일관성,가용성,분할내성 이 세가지 조건을 모두 만족하는 분산 컴퓨터 시스템이 존재하지 않음을 증명한 정리를 말한다.

일관성 : 모든 요청은 최신 데이터 또는 에러를 응답받는다

가용성 : 모든 요청은 정상 응답을 받는다

파티션 허용성(분할내성) : 노드간 통신이 실패하는 경우라도 시스템은 정상 동작한다.

일관성과 가용성을 동시에 완벽하게 만족하려면, 네트워크 장애를 허용하지 않아야한다. 네트워크 장애가 절대 일어나지 않는 네트워크 구성은 불가능하기 때문에 CAP 이론은 네트워크 파티션 허용을 전제하기 때문에, CA환경은 구성 불가능하다.

CAP 이론에 따르면 분산시스템은 CP이거나 AP 여야 한다. 그러나 실제 시스템은 둘 중에 하나라고 명확히 구분지을 수 없으며, 완벽한 CP, AP 시스템은 사실상 쓸모가 없다.

완벽한 CP 시스템은 하나의 트랜잭션이 다른 모든 노드에 복제된 후에 완료될 수 있다. 이런 시스템은 하나의 노드라도 문제가 있으면 트랜잭션은 무조건 실패하고, 노드가 많아지면 지연시간이 증가한다.

완벽한 AP 시스템은 모든 노드가 어떤 상황에서라도 응답할 수 있어야 한다. 하나의 노드가 네트워크 파티션으로 인해 고립되면, 그 고립된 노드가 가지고 있는 데이터는 쓸모없어지지만 응답을 할수있게만 만들어도 완벽한 가용성을 가지게 된다. 이 노드와 운나쁘게 연결된 사용자는 문제를 인지하지 못하고 계속해서 사용할 수 있다.

완벽한 CP, AP를 보면 반드시 둘 중 하나만을 선택해야하는 것은 아니라는 결론에 도달할 수 있다.

마지막으로 CAP의 경우 파티션이 없는 상황을 설명하지 못한다는 것이다. 파티션이 없는 상황에서도 분산 시스템은 상충하는 특성들이 있고, 장애상황만큼이나 정상 상황에서 시스템이 어떻게 동작하는지도 중요하다.

2. PACELC에 대해 간단히 설명하고, 해당 이론에서 도출할 수 있는 기준 4가지에 대해 설명하시오.

CAP 이론의 단점을 보완하기 위해 나온 이론으로, 네트워크 파티션 상황에서 일관성 - 가용성을 이용하여 시스템의 특성을 설명한다면, 거기에 정상 상황이라는 새로운 축을 더해 설명하는 이론이다.

PACELC 은 파티션 발생에 따라 다음과 같은 기준이 생성된다.

장애	정상	장애상황	정상상황
PA	EL	가용노드만 반영	지연시간 고려
PA	EC	가용 노드만 반영	모든 메세지 보장
PC	EL	Timeline 일관성 보장	지연시간 우선 고려
PC	EC	일관성 우선 보장	모든 메세지 보장

CPU 스케줄링

1. CPU 스케줄링이 발생하는 이유는 무엇일까요?

CPU를 프로세스간에 교환함으로써 자원을 보다 생산적으로 사용하기 위함이다. CPU 이용률을 최대화 하여 항상 실행중인 프로세스를 가지도록 하게 하기 위함 ! 프로세스가 대기해야할 경우 운영체제는 CPU를 해당 프로세스로부터 회수하여 다른 프로세스에 하기 위해 스케줄링을 하는 것이다 !

2. 비선점 스케줄링과 선점 스케줄링의 차이는 무엇일까요?

참고자료1

참고자료2

스케줄링은 네가지 상황에서 발생한다.

첫번째, 한 프로세스가 실행 상태에서 대기 상태로 전환될 때 (I/O발생)

두번째, 프로세스가 실행 상태에서 준비완료 상태로 전환될때 (인터럽트 발생)

세번째, 프로세스가 대기 상태에서 준비 완료 상태로 전환될 때 (I/O 종료)

네번째, 프로세스가 종료할 때

비선점 스케줄링의 경우 CPU가 한 프로세스에게 할당되면 프로세스가 종료하든지, 또는 대기 상태로 전환해 CPU를 방출할 때 까지 점유한다 (1,4)

선점 스케줄링의 경우 시분할 시스템에서 타임슬라이스가 소진되었거나, 인터럽트나 시스템 호출 종료시에 더 높은 우선 순위 프로세스가 발생 되었음을 알았을 때,현 실행 프로세스로부터 강제로 CPU를 회수하는 것을 말한다 (2,3)

타임슬라이스란 ? 시분할의 핵심으로 타임슬라이스에서 무조건 스케줄링을 진행하는 것을 말한다. 이에 따라 한 프로그램에서 입출력 완료 후 대기시간이 적어졌다. 타임슬라이스는 타이머 인터럽트를 통해 구현되며, 10ms 의 인터벌로 타이머 인터럽트를 설정해 구현된다. 이런 시분할 시스템을 통해 사용자 사이를 재빠르게 전환해줌으로서 사용자로 하여금 컴퓨터를 독점하고 있다고 생각하게 만든다.

CPU 소유권을 어떻게 가지느냐가 쟁점

3. 비선점형 알고리즘 중 한 가지와 선점형 알고리즘 중 한 가지를 선택해 각각의 특징을 짧게 요약해주세요.

참고자료1

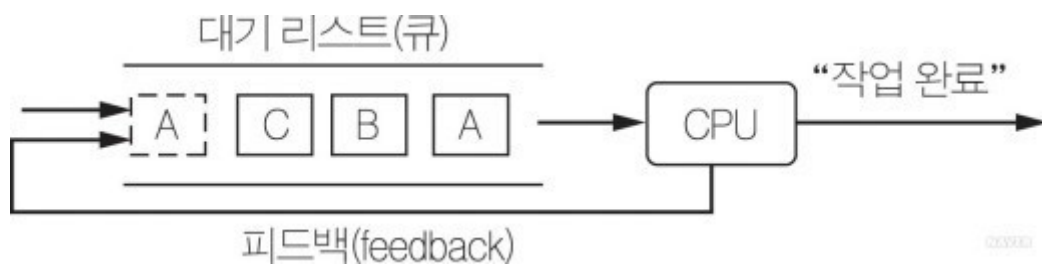
참고자료2

라운드로빈(선점형)

시분할 시스템을 위해 설계된 선점형 스케줄링의 하나로, 프로세스들 사이에 우선순위를 두지 않고, 순서대로 시간단위로 CPU를 할당하는 방식의 스케줄링 알고리즘이다.

풀어서 말하면, 컴퓨터 운영에서, 자원을 사용할 수 있는 기회를 프로그램 프로세스들에게 공정하게 부여하기 위한 한 방법으로, 각 프로세스에 일정시간을 할당하고, 할당된 시간이 지나면 그 프로세스는 잠시 보류한 뒤 다른 프로세스에게 기회를 주고, 또 그 다음 프로세스에게 하는 식으로 돌아가며 기회를 부여하는 운영방식이라고 말할 수 있다.

짧게 말하자면 가장 먼저 들어온 프로세스가 할당 받은 시간에만 실행 후 다음 프로세스가 시간을 할당받는 식으로 발생



10ms ~ 100ms 시간단위동안 수행한 프로세스는 준비 큐의 끝으로 밀려나게 되고, 문맥전환의 오버헤드가 큰 반면에 응답시간이 짧아지는 장점이 있어서 실시간 시스템에 유리하다.