1) ping 명령어로 서버를 점검한다?

ping 명령어로 서버가 죽었는지 살았는지 확인 할 수 있다. 물론 서버나 네트웍 장비에서 ping을 막는다면, 불가능 한 일이지만, 그렇지 않다는 가정하에서 ping 명령어를 이용하겠다.

(1) 서버 점검
ping 아이피

위와같이 명령어를 입력하면, 정상적 이라면, 지속적으로 응답시간을 보여 주게된다.

(2) ping을 한번만 테스트 하려면?
ping -c 1 아이피

이렇게 하면, ping 테스트를 한번만 한다. 정상적일 때는 응답이 빠른데 실패는 너무 느리다.

(3) ping 테스트시 응답을 기다리는 시간을 줄이려면?
ping -c 1 -w 1 아이피

위와같이 하면, 응답이 없더라도 1초만 기다리게 된다.

2) 서버의 응답 결과를 쉘에서 처리하자!

ping 명령의 결과는 사용자가 보기 좋은 형태로 나타난다. 하지만, 프로그램을 작성하기 위해서는 단순하게 "된다" 혹은 "안된다" 정도가 편리하다. 모든 명령어는 명령어 실행 결과를 리턴하게 되어 있다. 대부분 정상적으로 작동하였으면 0 , 아니면 에러코드 를 리턴한다.

bash쉘에서 바로 전 명령어의 리턴값을 받아오는 변수가 있다 "$?" 이다.
이 변수를 가지고 다음과 같이 ping 명령의 결과를 처리 할 수 있다.

```
ping -c 1 -w 1 아이피 &> /dev/null
if [ "$?" == "0" ] ; then
   echo "정상"
else
   echo "비정상"
fi
```

위 명령으로 ping 결과에 대한 리턴값을 받게 되며, 결과값이 0이면 "정상"을 출력하고 아니면, "비정상"을 출력하게 된다.
ping 명령어의 출력은 /dev/null으로 보내 버린다

3) 서버 리스트를 입력하자
내 서버가 여러대라면, 서버 리스트를 입력해야 한다. 여기서는 쉘스크립트의 배열을 사용하도록 한다.

(1) 배열을 입력한다.

```
server[0]="192.168.0.100"
server[1]="192.168.0.1"
```

위와같이 배열으로 선언했다.
server 라는 배열에 0번째 주소에서는 "192.168.0.100"
server 라는 배열에 1번째 주소에서는 "192.168.0.100"

서버가 많이 있다면, 많이 넣어주면 된다.^^

(2) 배열의 내용을 보여준다.

```
echo ${server[0]}
```

위와같이 출력할 수 있다.

(3) 배열의 크기가 궁금하다?

```
echo ${#server[*]}
```

(4) 배열의 모든 내용을 출력하려면?

```
echo ${server[*]}
```

우리는 이렇게 서버 리스트를 배열에 넣었고 데이터를 확인 했다.

4) '3)'에서 넣었던 데이터를 '2)'의 ping 명령으로 점검해 보도록 하자!
이제 반복문 인 for 문을 알아 보자. #3 강좌에서는 for in 문을 배웠지만, 지금 사용하는 for 문은 일반적인 언어에 있는 for 문이다. c 언어의 그것과 비슷하니 살펴 보도록 하자.

```
for (( 표현식1 ; 표현식2 ; 표현식3 )) ; do
  <명령어들>
done
```

일반적인 언어와 같이 "표현식1"은 초기에 1번만 실행된다.
"표현식2"는 조건문이 들어간다.
"표현식3"은 증감문이 들어간다.

다음 예제를 보자

```
for (( i=1 ; i<=10 ; i++ )) ; do
  echo "$i"
done
```

위 예는 1부터 10까지 1씩 증가하면서 출력하는 프로그램이다.

그렇다면, '3)'에서 넣었던 서버리스트를 '2)'의 명령어로 점검해 보자.

```
for (( i=0 ; i<${#server[*]} ; i++ )) ; do
  ping -c 1 -w 1 ${server[$i]}  &> /dev/null
  if [ "$?" == "0" ] ; then
    echo "${server[$i]}  .. 정상"
  else
    echo "${server[$i]}  .. 비정상"
  fi
done
```

위와같이 하면, 0번부터 배열의 크기만큼 반복해서 ping 테스트를 하고,
결과를 서버 IP와 함께 정상 / 비정상 보여 준다.

5) 우리의 목적을 달성해 보자..
위에서 알아본 것들을 응용해서 다음과 같이 우리가 원하는 쉘스크립트를 만들어 보자.

```
server[0]="192.168.0.100"
server[1]="192.168.0.1"
```

```
for (( i=0 ; i<${#server[*]} ; i++ )) ; do
  ping -c 1 -w 1 ${server[$i]} &> /dev/null
  if [ "$?" == "0" ] ; then
    echo "${server[$i]}  .. 정상"
  else
    echo "${server[$i]}  .. 비정상"
  fi
done
```

위 쉘스크립트를 이용하여 간단하게 서버가 죽었는지 살아 있는지 점검 할 수 있다.

출처 :
https://www.linux.co.kr/home/lecture/index.php?cateNo=&secNo=&theNo=&leccode=11061