목차

2021년 1월 4일 월요일 오후 3:22

20.12.28

2020년 12월 28일 월요일 오후 2:51

리눅스를 배워야하는 이유

1. 데이터 분석가가 갖추어야할 기본 기술 중에 하둡 사용이 있는데 하둡이 리눅스에 기반을 두고 있기 때문에 리눅스를 먼저 배워야 한다.

2. 딥러닝을 구현할때 현업에서 많은 부분이 리눅스에 파이썬을 사용해서 구현한다.

3. 빅데이터 기사 시험에 하둡이 나오는데 여러 용어들에 대한 정확한 이해가 있어야 한다.

4. 딥러닝으로 무언가를 만들 때 리눅스가 더 편리하다.

리눅스의 유래

https://www.youtube.com/watch?v=qmLBJ6sTing

리눅스란 유닉스(unix)가 너무 고가여서 리눅스 오픈 소스를 핀란드의 리누즈 토발즈 학생이 1991년 11월에 개발 하였다.
리누즈 토발즈가 리눅스 커널(자동차의 엔진과 같음)을 개발하고 소스를 무료로 공개했다.
그리고 전세계의 개발자들이 이 오픈 소스를 가져다가 더 좋게 개선해서 인터넷에 올리는 작업을 반복하다 보니 리눅스 os가 유닉스보다 더 가볍고 안정적이게 되었다.

GNU 프로젝트 : 누구든지 배포된 오픈소스를 가져다가 개발 할 수 있고 돈을 벌 목적으로 상용화를 할 수도 있는데 소스를 가져다가 더 좋게 수정하면 그 코드를 인터넷에 올려줘야 한다.

리눅스의 종류

1. Oracle linux
2. Cent os
3. Ubunt

리눅스 설치

http://cafe.daum.net/oracleoracle/Sfno/1

1. Oracle virtual box 설치 (가상의 컴퓨터를 내 컴퓨터 안에 만든다.)

2. cenos 7버전 다운로드

- CentOS-7-x86_64-DVD-2009.iso 다운로드

20.12.29

2020년 12월 29일 화요일 오전 9:45

게스트 cd 확장 설치

- 리눅스와 가상서버를 이동하면서 마우스와 키보드를 편하게 이용하기 위해 작업한다

1. root 로 접속한다.
2. 게스트 cd를 cd룸에 입력한다.
3. 게스트 cd를 실행한다.
4. 리눅스 시스템을 reboot 한다.

리눅스 시스템 인터넷 설정

1. 리눅스의 시작 버튼
2. system tools 의 settings
3. network 텝을 클릭
4. ethernet(enp0s3) 을 on 한다.
5. firefow 를 키고 naver로 접속

리눅스 기본 명령어

1. **cd 명령어**

- Change Directory 명령어로 디렉토리를 이동하는 명령어이다.

  ex)
  # whoami : 접속한 내가 누구인지 확인하는 명령어
  # pwd : 현재 내가 있는 디렉토리를 확인하는 명령어 (print working directory)
  # ls : list명령어로 현재 디렉토리에 있는 폴더와 파일을 확인
  # cd Documents : Documents 디렉토리로 이동
  # cd .. : 뒤로 이동 (전 디렉토리로 이동)

문제1. 현재 디렉토리에서 Desktop 디렉토리로 이동하시오.

```
# cd Desktop
# pwd
```

문제2 . 다시 뒤로 이동하시오.

```
# cd ..
```

```
# pwd
```

문제3. 나의 집으로 가시오.

```
# whoami
# cd
# pwd
```

cd : 내가 어디에 있든 집으로 돌아간다.

경로(path)에는 크게 2가지 경로가 있다.

1. 절대경로 : cd 내가 가고자 하는 정확한 위치
   ex)
   # cd /root/Desktop

2. 상대경로 : 나의 현재위치를 상대로 이동한다.
   ex)
   # cd ..
   - 현재위치를 상대로 상위 디렉토리로 이동

문제4. 집에서 test01 디렉토리를 생성하시오.

```
# cd
# mkdir test01
# ls
```

문제5. test01 폴더(디렉토리)로 이동하시오.

```
# cd test01
# pwd
```

or

```
# cd /root/test01
# pwd
```

문제6. 집에서 test02 디렉토리를 생성하고 이동하시오.

```
# cd
# mkdir test02
# ls
# cd test02
# pwd
```

문제7. 집에서 test03 디렉토리를 생성하고 test03 디렉토리로 가서 test04 디렉토리를 생성하고 test04 디렉토리로 이동하시오.

```
# cd
# mkdir test03
# ls
# cd test03
# pwd
# mkdir test04
# cd test04
# pwd
```

문제8. 집에서 test03 밑에 test04로 이동하시오.

```
# cd
# cd test03/test04
# pwd
```

문제9. 위에서 이동한 디렉토리에서 방금 이동하기 전 디렉토리로 바로 이동하시오.

```
# cd -
# pwd
```

문제10. 집에서 test07 디렉토리를 생성하고 test07 디렉토리에 test08 디렉토리를 만들고 test08 디렉토리에 test09 디렉토리를 만들고 test09 디렉토리로 이동하시오.

```
# cd
# mkdir test07
# cd test07
# mkdir test08
# cd test08
# mkdir test09
# cd test09
# pwd
```

2. **touch 명령어**
   - 파일의 용량이 0인 파일을 생성하는 명령어

ex)

```
# touch a1.txt
# ls -l a1.txt
```

a1.txt의 용량을 확인한다.

문제11. 집에서 아래의 파일들의 크기를 0으로 해서 생성하시오.

a.txt, b.txt, c.txt, d.txt, e.txt, f.txt

```
# cd
# touch a.txt b.txt c.txt d.txt e.txt f.txt
# ls -l *.txt
```

ls -l *.txt
확장자가 .txt인 파일들만 보여준다.

3. **mkdir 명령어 (make directory의 약자)**
   - 디렉토리를 만드는 명령어

매뉴얼을 보는 명령어
whatis mkdir
man mkdir

문제12. 집에서 아래의 디렉토리들을 생성하시오.

/root/test30/test31/test32/test33/test34/test35

```
# cd
# mkdir -p /root/test30/test31/test32/test33/test34/test35
```

-p 옵션은 디렉토리를 만들면서 한번에 하위 디렉토리를 생성하는 옵션이다.

문제13. 집에서 파이썬 스크립트를 따로 관리할 디렉토리를 ptest01 로 생성하고 ptest01 디렉토리로 이동해서 파일의 크기가 0인 example01.txt 파일을 생성하시오.

```
# cd
# mkdir ptest01
# cd ptest01
# touch example01.txt
# pwd
# ls
```

4. **rm 명령어**
   - 파일이나 디렉토리를 삭제하는 명령어

리눅스나 유닉스는 삭제할 때 특히 주의해야 한다.

ex)
touch bbb.txt
ls -l bbb.txt
rm bbb.txt
ls -l bbb.txt

문제14. 집으로 이동해서 현재 디렉토리에 있는 a.txt, b.txt, c.txt를 삭제하시오.

```
# cd
# rm a.txt b.txt c.txt
# ls -l *.txt
```

문제15. 집으로 이동해서 test45라는 디렉토리를 생성하시오.

```
# cd
# mkdir test45
# ls -ld test45
```

-ld 옵션을 사용하면 test45 디렉토리의 정보만 따로 확인 할 수 있다.

문제16. 집으로 이동해서 test45 디렉토리를 삭제하시오.

```
# cd
# rm -r test45
```

-r 옵션은 디렉토리를 삭제할 때 사용하는 옵션이다.
디렉토리 삭제시 삭제할 디렉토리 밑에 있는 파일과 하위 디렉토리를 모두다 삭제하는 옵션이다.

문제17. 집으로 이동해서 test03 디렉토리 밑에 하위 디렉토리가 있는지 확인하시오.

```
# cd
# cd test03
# ls
# cd test04
# ls
```

문제18. 집으로 이동해서 test03 디렉토리와 그 하위 디렉토리를 모두

삭제하시오.

```
# cd
# rm -r test03
# ls -ld test03
```

5. **rmdir 명령어**
   - rmdir 명령어는 디렉토리를 삭제하는 명령어

ex)
```
# mkdir test50
# ls -ld test50
# rmdir test50
# ls -ld test50
```

문제19. 집으로 이동해서 /root 밑에 test51 디렉토리를 생성하고 test51 디렉토리에 emp.txt 를 올리시오.

장치 > 클립보드 공유 > 양방향
장치 > 드래그 앤 드롭 > 양방향

```
# cd
# mkdir test51
```

6. **alias 명령어**
   - 자주 수행하는 명령어들을 쉽게 사용할 수 있도록 설정하는 명령어

ex)
```
# pwd
# alias p='pwd'
# p
```

pwd라고 다 쓰지 않고 p만 써서 수행한다.

문제20. ls -l 명령어를 쉽게 사용할 수 있도록 l 로 설정하시오.

```
# alias l='ls -l'
# l
```

문제21. 다시 p alias를 지우시오.

```
# unalias p
```

문제22. alias로 등록된 목록을 확인하시오.

alias

문제23. l로 만들었던 alias 를 지우시오.

unalias l

7. **cat 명령어**
   - 파일의 내용을 화면에 출력하는 명령어

ex)
cat emp.txt

문제24. 스티븐 잡스 연설문 (jobs.txt)을 /root 밑에 올리시오.

문제25. jobs.txt를 cat 으로 여시오.

cd
cat jobs.txt

문제26. jobs.txt를 more 명령어로 여시오.

cd
more jobs.txt

8. **redirection 명령어**
   - 화면에 출력되는 결과를 파일로 저장하는 명령어

>> : 없으면 파일을 생성하고 있으면 기존 파일뒤에 덧붙인다.
> : 파일을 생성한다. 기존의 파일이 있으면 그냥 덮어 씌운다.

cat emp.txt >> emp2.txt
ls -l emp*.txt
cat emp2.txt

cat emp.txt로 본 화면의 결과를 emp2.txt로 저장한다.

문제27. jobs.txt를 cat 으로 열어서 본 결과를 jobs2.txt로 저장하시오.

cat jobs.txt >> jobs2.txt
ls -l *.txt
cat jobs2.txt

문제28. 집에서 ls -l 로 본 결과를 list.txt.로 저장하시오.

```
# cd
# ls -l >> list.txt
# cat list.txt
```

9. **more 명령어**
   - 1 페이지가 넘는 문서의 내용을 화면에 출력할 때 페이지 단위로 볼 수 있는 명령어

ex)
more jobs.txt

전진키 : 스페이스
후진키 : b
페이지 단위로 전진 : f

10. **head 명령어**
    - 문서의 처음 몇줄을 화면에 출력하는 명령어

ex)
head 출력줄수 파일명
head -5 jobs.txt

문제29. jobs.txt 의 위의 10 줄을 jobs_10.txt로 저장하시오.

```
# head -10 jobs.txt >> jobs_10.txt
# cat job_10.txt
```

11. wc 명령어
    - 파일 안에 단어의 개수 또는 라인수를 출력하는 명령어

ex)
wc -l 파일명
wc -l jobs.txt

옵션	설명
-l	라인수
-w	단어의 개수
-c	철자의 개수

문제30. 집으로 이동해서 확장자가 .txt인 파일을 조회하시오.

```
# cd
# ls -l *.txt
```

문제31. 위에 출력되는 결과에 라인수를 출력하시오.

```
# ls -l *.txt >> list2.txt
# wc -l list2.txt
```

문제32. 위에 작업을 파일을 생성하지 않고 더 간단하게 하시오.

```
# ls -l *.txt | wc -l
```

11. **pipe 명령어 (|)**
    - 앞의 명령어의 출력을 뒤의 명령어의 입력으로 보냄으로써 명령어의 실행결과를 다음 명령어로 전달하는 기능

문제33. /root 밑에 디렉토리가 몇 개 있는지 조회 하시오.

```
# cd
# ls -ld */ | wc -l
```

ls -ld */ 라고 하면 현재 디렉토리에 있는 디렉토리 리스트들만 출력하고 이 출력 결과를 파이프 다음 명령어의 입력으로 전달한다. 파이프 다음 명령어가 wc -l의 입력으로 전달하면 그 결과의 라인수를 출력한다.

12. **grep 명령어**
    - 파일 안에 포함된 특정 단어나 구문을 검색하는 명령어

ex)
```
# grep '찾고싶은 단어' 파일명
# grep 'SCOTT' emp.txt
```

위의 결과를 보면 emp.txt 에서 SCOTT이 포함된 행만 출력한다.

ex)
```
# grep -i 'scott' emp.txt
```

-i 옵션은 대소문자를 구분하지 않도록 해준다.

문제34. 직업이 SALESMAN인 사원들의 모든 행을 출력하시오.

```
# grep -i 'salesman' emp.txt
```

문제35. 직업이 SALESMAN인 사원들은 전부 몇명인지 출력하시오.

```
# grep -i 'salesman' emp.txt | wc -l
```

문제36. 직업이 SALESMAN인 사원들의 이름과 월급을 출력하시오.

```
# grep -i 'salesman' emp.txt | awk '{print $2, $6}'
```

awk 명령어는 열을 선택하는 명 이다. $2 emp.txt 에서 두번째 열 이고 열과 열은 공백으로 구분되어져 있다.

문제37. 직업이 ANALYST인 사원들의 이름과 월급과 부서번호를 출력하시오.

```
# grep -i 'analyst' emp.txt | awk '{print $2, $6, $8}'
```

문제38. http://cafe.daum.net/oracleoracle/SfZF/1324 를 참고해서 신뢰구간 문제를 푸는데 동전을 10번 던져서 뒷면이 0번 ~ 10번 나오는 횟수로 신뢰구간 95%에 속하는지의 여부를 출력되게 하시오.

(파이썬 문제)

```
print (coin_hypo(4))
```

동전을 10 번 던졌을 때 뒷면이 나오는 횟수가 4번이 나올 확률은 신뢰구간 95%안에 있습니다.

```
print (coin_hypo(10))
```

동전을 10 번 던졌을 때 뒷면이 나오는 횟수가 10번이 나올 확률은 신뢰구간 95%안에 없습니다.

```
import random
import numpy as np

def coin_avg_std(num): # 동전을 몇번 던지냐에 따른 평균과 표준편차 를 생성
    result=[]
    for i in range(num):
        cnt = 0
        for t in range(10):
            cnt += (random.randint(0,1))
```

```
        result.append(cnt)
    c_m = np.mean(result)
    c_s = np.std(result)
    return c_m, c_s

def coin_hypo(num):
    c_m,c_s = coin_avg_std(10000) # 동전 몇번 던진지를 선택
    if c_m-1.96*c_s<=num<=c_m+1.96*c_s:
        return f'동전을  10번 던졌을 때 뒷면이 나오는 횟수가 {num}번이 나올 확률은  신뢰구간 95%안에 있습니다.'
    else:
        return f'동전을  10번 던졌을 때 뒷면이 나오는 횟수가 {num}번이 나올 확률은  신뢰구간 95%안에 없습니다.'

print(coin_hypo(4))
print(coin_hypo(10))
```

20.12.30

2020년 12월 30일 수요일 오전 9:43

문제39. 이름이 allen 인 사원의 이름과 월급을 출력하시오.

```
# grep -i 'allen' emp.txt | awk '{print $2, $6}'
```

문제40. 월급이 3000인 사원의 이름과 월급을 출력하시오.

```
# grep '3000' emp.txt | awk '{print $2, $6}'
```

위처럼 emp.txt 만 이용하면 grep만으로 위의 문제를 해결할 수 있지만 월급이 3000인 데이터를 정확하게 검색하려면 awk를 이용해야 한다.

문제41. 부서번호가 10번인 사원들의 이름과 월급과 부서번호를 출력하시오.

```
# grep -i '10' emp.txt | awk '{print $2, $6, $8}'
```

위와 같이 검색을 하면 월급 1100에 10 이 포함되어 있는 ADAMS 1100 20 데이터도 같이 출력된다.

-w 옵션은 단어별 검색이다.

```
# awk '{print $2, $6, $8}' emp.txt | grep -iw '10'
```

awk 로 먼저 emp.txt 에서 이름과 월급과 부서번호를 선택해서 출력하고 그 출력된 결과에서 10을 포함하고 있는 행들만 출력한다.

grep은 emp.txt에서 특정 글자가 포함되어져 있는 행들만 검색하기 때문에 특정 열에 대해서 데이터를 검색하는것은 grep만으로는 어렵기 때문에 awk 명령어를 사용해야 한다.

13. awk 명령어

- 특정 컬럼을 출력하고자 할 때 사용하는 명령어

ex)
awk '패턴 {action}' 대상 파일명
awk '$3 == "SALESMAN" {print $2, $3}' emp.txt

직업이 SALESMAN인 사원의 이름과 직업을 출력하는 명령어

리눅스 연산자 3가지 정리

1. 산술 연산자 : +, -, *, /
2. 비교 연산자 : >, <, >=, <=, ==, !=
3. 논리 연산자 : &&, ||, !

문제42. 월급이 3000 이상인 사원들의 이름과 월급을 출력하시오.

```
# awk '$6 >= 3000 {print $2, $6}' emp.txt
```

문제43. 직업이 salesman이 아닌 사원들의 이름과 직업을 출력하시오.

```
# awk '$3 != toupper("salesman") {print $2, $3}' emp.txt
```

문제44. 직업이 salesman 이고 월급이 1500 이상인 사원들의 이름과 월급과 직업을 출력하시오.

```
# awk '$3 == toupper("salesman") && $6 >= 1500 {print $2, $6, $3}' emp.txt
```

문제45. 81년도에 입사한 사원들의 이름과 입사일을 출력하시오.

오라클의 substr 과 같은 기능이 있는지 확인

```
# man awk
```

/substr

- 페이지에서 substr을 검색하는 명령어

```
# awk 'substr($5, 3, 2) == "81" {print $2, $5}' emp.txt
```

문제46. 12월에 입사한 사원들의 이름과 입사일을 출력하시오.

```
# awk 'substr ($5, 6, 2) == "12" {print $2, $5}' emp.txt
```

문제47. 81년도에 입사하지 않은 사원들의 이름과 입사일을 출력하시오.

```
# awk 'substr ($5,3,2) != "81" {print $2, $5}' emp.txt
```

문제48. 이름의 첫글자가 A 로 시작하는 사원들의 이름과 월급을 출력하시오.

```
# awk '{print $2, $6}' emp.txt | grep -i '^a'
```

^ : 시작, $: 종료

문제49. 이름의 끝 글자가 T 로 끝나는 사원들의 이름과 월급을 출력하시오.

```
# awk '{print $6,$2}' emp.txt | grep -i 't$'
```

문제50. 위의 문제를 substr로 출력하시오.

```
# awk 'substr($2,5,1) == "T" {print $2, $6}' emp.txt
```

문제51. 이름의 두번째 철자가 M인 사원들의 이름과 월급을 출력하시오.

```
# awk 'substr($2,2,1) == "M" {print $2, $6}' emp.txt
```

14. sort 명령어

- data를 특정 컬럼을 기준으로 정렬하는 명령어

ex)
sor 옵션 파일명
sort -nk 6 emp.txt

-n 옵션을 사용하면 숫자로 변환해서 정렬을 해준다.
-k 옵션을 사용하고 6을 쓰면 6번째 컬럼을 기준으로 정렬 해준다.

sort -nrk 6 emp.txt

-r 옵션을 사용하면 월급이 높은 사원부터 출력된다.

문제52. 이름과 월급을 출력하는데 월급이 높은 사원부터 출력하시오.

```
# sort -nrk 6 emp.txt |awk '{print $2,$6}'
```

문제53. 직업이 SALESMAN인 사원들의 이름과 월급과 직업을 출력하는데 원급이 높은 사원부터 출력하시오.

```
# awk '$3 == toupper("salesman") {print $2,$6,$3}' emp.txt | sort -nrk 2
```

문제54. 직업이 SALESMAN이 아닌사원들의 이름과 월급과 직업을 출력하는데 월급이 낮은 사원부터 출력하시오.

```
# awk '$3 != toupper("salesman") {print $2,$6,$3}' emp.txt | sort -nk 2
```

문제55. 사원테이블의 월급의 토탈값을 출력하시오.

```
# awk '{print $6}' emp.txt | awk '{sum += $1} END {print sum}'
```

월급만 추출해서 그 출력값을 파이프 다음에 입력값으로 보낸다.
sum += $1 이 실행되면서 월급이 계속 누적이 된다.
END로 종료되면 sum을 프린트 한다.

문제56. 직업이 SALESMAN인 사원들의 토탈월급을 출력하시오.

```
# grep -i 'salesman' emp.txt | awk '{print $6}' | awk '{sum += $1} END {print sum}'
```

문제57. 직업이 SALESMAN이 아니고 월급이 2000 이상인 사원들의 이름과 월급과 직업을 출력하시오.

```
# awk '$3 != toupper("salesman") && $6 >= 2000 {print $2,$6,$3}' emp.txt
```

문제58. 위의 사원들의 월급의 토탈값을 출력하시오.

```
# awk '$3 != toupper("salesman") && $6 >= 2000 {print $2,$6,$3}' emp.txt | awk '{sum += $2} END{print sum}'
```

문제59. 81년도에 입사한 사원들의 월급의 토탈값을 출력하시오.

```
# awk 'substr($5,3,2) == "81"' emp.txt | awk '{sum += $6} END {print sum}'
```

문제60. 아래의 SQL의 결과를 리눅스 명령어로 출력하시오.

```
select ename || '의 월급은' || sal || '입니다.'
        from emp;
```

```
# awk '{print $2 "의 월급은" $6 "입니다."}' emp.txt
```

문제61. 아래의 SQL의 결과를 리눅스 명령어로 출력하시오.

```
select ename || '의 직업은' || job || '입니다.'
        from emp;
```

```
# awk '{print $2 "의 직업은" $3 "입니다."}' emp.txt
```

문제62. 집에서 ls -l 로 수행한 결과를 출력하시오.

```
# cd
# ls -l
```

문제63. 위에서 출력된 결과에서 파일의 크기에 해당하는 부분만 출력하시오.

```
# cd
# ls -l | awk '{print $5}'
```

문제64. 위에서 출력된 숫자들의 총합을 출력하시오.

```
# ls -l | awk '{print $5}' | awk '{sum += $1} END {print sum}'
```

15. uniq 명령어

- 중복된 라인을 제거하는 명령어

ex)

uniq 옵션 파일명

문제65. emp.txt에서 직업만 추출하시오.

```
# awk '{print $3}' emp.txt
```

문제66. 위의 결과를 abc 순으로 정렬해서 출력하시오.

```
# awk '{print $3}' emp.txt | sort
```

문제67. 위의 결과를 중복제거해서 출력하시오.

```
# awk '{print $3}' emp.txt | sort | uniq
```

위아래가 같은 데이터만 중복을 제거하기 때문에 uniq를 사용하기 전에 정렬을 해야 한다.

문제68. 사원 테이블에서 부서번호를 출력하는데 중복을 제거해서 출력하시오.

```
# awk '{print $8}' emp.txt | sort | uniq
```

문제69. 직업이 CLERK인 사원들의 부서번호를 출력하는데 중복 제거해서 출력하시오.

```
# awk '$3 == toupper("clerk") {print $8}' emp.txt | sort | uniq
```

16. echo 명령어

- 출력하고자 하는 글자를 출력할 때 사용하는 명령어

파이썬의 print 와 같은 명령어 이다.
변수 안의 글자를 출력할 때 사용하는 명령어

ex)

```
# a=1
# echo $a
# b='scott'
# echo $b
```

변수 안에 있는 값을 출력할 때는 $를 앞에 붙여줘야 한다.

문제70. 직업을 물어보게하고 직업을 입력하면 해당 직업을 갖는 사원들의 이름과 직업이 출력되게 하시오.

```
# vi job2.sh

echo -n 'Enter the job name'
read job
grep -i $job emp.txt | awk '{print $2, $3}'

# sh job2.sh
```

-n 은 엔터와 같은 역활을 한다.

위의 리눅스 명령어 여러개를 파일로 저장해서 한번에 수행하는 방법

vi 편집기 화면으로 들어가서 위의 세줄을 복사해서 붙여 넣으려면 알파벳 i 를 눌러 아래쪽에 insert 가 나오는지 확인 후 붙여넣고 저장해야 하는데 저장하려면 esc 키를 눌러 아래쪽의 insert를 사라지게 하고 대문자 Z를 두번 연속으로 눌러서 저장하고 빠져나오면 된다.

sh job2.sh
.sh 는 쉘스크립트이다.
쉘스크립트는 리눅스 명령어를 모아놓은 것이다.

쉘스크립트를 잘 작성할 줄 알면 업무의 많은 부분을 자동화 할 수 있다.

sh 명령어로 쉘스크립트를 실행한다.

문제71. 아래와 같이 부서번호를 물어보게 하고 부서번호를 입력하면 해당 부서번호인 사원들의 이름과 월급과 부서번호가 출력되게 하시오.

```
# vi deptno.sh

echo -n 'Enter the deptno'
read deptno
awk -v num=$deptno '$8==num {print $2,$6,$8}' emp.txt

# sh deptno.sh
```

awk의 -v 옵션은 변수의 값을 다른 변수에 할당할 때 사용하는 옵션이다.

문제72. 이름이 SCOTT인 사원의 이름과 월급을 출력하시오.

```
# awk '$2==toupper("scott") {print$2, $6}' emp.txt
```

문제73. 이름을 물어보게 하고 이름을 입력하면 해당사원의 이름과 월급이 출력되게 하시오.

```
# vi find_sal.sh

echo -n 'Enter the ename'
read ename
awk -v name=$ename '$2==toupper(name) {print $2,$6}' emp.txt

# sh find_sal.sh
```

문제74. dept.txt를 리눅스 서버에 올리시오.

문제75. dept.txt에서 부서번호가 10번의 부서위치가 어디인지 출력하시오.

```
# awk '$1 == 10 {print $3}' dept.txt
```

문제76. scott이 근무하는 부서위치를 출력하시오.

```
select d.loc
        from emp e, dept d
        where e.deptno = d.deptno
                and e.ename = 'SCOTT';

deptno=`awk '$2 == toupper("scott") {print $8}' emp.txt`
awk -v num=$deptno '$1==num {print $3}' dept.txt
```

양쪽 역따옴표 안쪽에 있는 명령어로 수행한 결과를 deptno변수에 할당한다.

문제77. 위의 스크립트를 이용해서 shell script를 생성해서 아래와 같이 실행되게 하시오.

```
# vi find_loc.sh

echo -n 'Enter the ename'
read ename
deptno=`awk -v name=$ename '$2 == toupper(name) {print $8}' emp.txt`
awk -v num=$deptno '$1==num {print $3}' dept.txt

# sh find_loc.sh
```

문제78. 직업이 SALESMAN인 사원들의 월급들을 출력하시오.

```
# awk '$3 == toupper("salesman") {print $6}' emp.txt
```

문제79. 위의 결과를 다시 출력하는데 월급 앞에 이름도 같이 출력하시오.

```
# awk '$3 == toupper("salesman") {print $2,$6}' emp.txt
```

문제80. 위의 스크립트를 이용해서 shell script를 만드는데 직업을 물어보게 하고 직업을 입력하면 해당 직업인 사원들의 이름과 월급이 출력되게 하시오.

```
# vi find_job.sh

echo -n 'Enter the job'
read job
awk -v num=$job '$3==toupper(num) {print $2,$6}' emp.txt

# sh find_job.sh
```

문제81. 문제80번 코드를 수정해서 직업을 물어보게 하고 직업을 입력하면 해당 직업의 사원들의 모든 데이터가 다 출력되게 하시오.

```
# vi find_job.sh

echo -n 'Enter the job'
read job
awk -v num=$job '$3==toupper(num) {print}' emp.txt

# sh find_job.sh
```

문제82. 직업을 물어보게하고 직업을 입력하면 해당 직업인 사원들의 모든 데이터로 텍스트 파일이 생성되게 하시오.

```
# vi find_job.sh

echo -n 'Enter the job'
read job
awk -v num=$job '$3==toupper(num) {print}' emp.txt  >>  $job.txt

# sh find_job.sh
```

문제83. 부서번호를 물어보게 하고 부서번호를 입력하면 해당 부서번호의 모든 사원의 데이터가 텍스트 파일로 생성되게 하시오.

```
# vi find_deptno.sh

echo -n 'Enter the deptno'
read deptno
awk -v num=$deptno '$8==num {print}' emp.txt >> $deptno.txt

# sh find_deptno.sh
```

20.12.31

2020년 12월 31일 목요일 오전 9:41

문제84. 이름이 SCOTT인 사원의 부서위치를 출력하시오.

1. scott 의 부서번호를 emp.txt에서 알아낸다.
2. scott 의 부서번호로 부서위치(loc)를 dept.txt에서 알아낸다.

```
# deptno=`awk '$2==toupper("scott") {print $8}' emp.txt`
# loc=`awk -v num=$deptno '$1==num {print $3}' dept.txt`t
# echo $loc
```

문제85. 위의 스크립트를 이용해서 이름을 물어보게하고 이름을 입력하면 해당 사원이 근무하는 부서위치가 출력되게 쉘 스크립트를 작성하시오.

```
# vi find_loc.sh

echo -n 'Enter the ename'
read ename
deptno=`awk -v name=$ename '$2 == toupper(name) {print $8}' emp.txt`
awk -v num=$deptno '$1==num {print $3}' dept.txt

# sh find_loc.sh
```

문제86. dept.txt에서 DALLAS의 부서번호를 출력하시오.

```
# awk '$3==toupper("dallas") {print $1}' dept.txt
```

문제87. 위의 부서번호를 가지고 emp.txt에서 해당 부서번호에서 근무하는 사원들의 이름과 직업을 출력하시오.

```
# deptno=`awk '$3==toupper("dallas") {print $1}' dept.txt`
# awk -v num=$deptno '$8==num {print $2,$3}' emp.txt
```

awk의 -v 옵션은 변수에 있는 값을 다른 변수에 담을 때 사용하는 옵션이다.

문제88. 위의 스크립트를 이용해서 부서위치를 물어보게 하고 부서위치를 입력하면 해당 부서위치인 사원들의 이름과 직업이 출력되게 하시오.

```
# vi find_loc2.sh

echo  -n  'Enter the loc:  '
read loc
deptno=`awk -v num=$loc '$3==toupper(num)  {print $1}'  dept.txt`
awk -v num=$deptno '$8==num {print $2,$3}'  emp.txt

# sh find_loc2.sh
```

문제89. 위의 스크립트를 이용해서 부서위치를 물어보게하고 부서위치를 입력하면 해당 부서위치에 속하는 사원들의 모든 데이터를 아래와 같이 생성되게 하시오.

```
# vi find_loc3.sh

echo  -n  'Enter the loc:  '
read loc
deptno=`awk -v num=$loc '$3==toupper(num)  {print $1}'  dept.txt`
awk -v num=$deptno '$8==num {print}'  emp.txt >> $loc.txt

# sh find_loc3.sh
```

17. diff 명령어

- 두 파일간의 차이점을 찾아서 알려주는 명령어

ex)
diff emp.txt dallas.txt

문제90. find_deptno.sh를 실행해서 20.txt 를 생성하시오.

```
# sh find_deptno.sh
```

문제91. emp.txt와 20.txt의 차이를 확인하시오.

```
# diff emp.txt 20.txt
```

18. find 명령어

- 검색하고자하는 파일을 찾을 때 사용하는 명령어

ex)
find 디렉토리 -name 파일명 -print

find /root -name 'emp.txt' -print

/root 디렉토리 밑에 emp.txt라는 파일이 있는지 검색하시오.

문제92. 집에서 test100 이라는 디렉토리를 만들고 test100 디렉토리 밑에 emp.txt 를 복사하시오.

```
# cd
# mkdir test100
# cp /root/emp.txt /root/test100/emp.txt
# find /root -name 'emp.txt' -print
```

/root/emp.txt를 /root/test100/emp.txt로 복사

문제93. /root 밑에 emp.txt가 있는지 find 하시오.

```
# find /root -name 'emp.txt' -print
```

문제94. 위의 문제93번을 다시 수행하는데 /root 밑에 하위디렉토리는 검색하지 않고 /root 밑에 있는 emp.txt 만 검색하시오.

```
# find /root -maxdepth 1 -name 'emp.txt' -print
```

문제95. 위의 명령어를 이용해서 아래와 같이 검색을 편하게 할 수 있도록 shell 스크립트를 만드시오.

```
Enter the file name:
Enter the maxdepth:
```

```
# vi find_file.sh
```

```
echo -n 'Enter the file name: '
read file_name
echo -n 'Enter the maxdepth: '
read max_depth
find /root -maxdepth $max_depth -name $file_name -print
```

```
# sh find_file.sh
```

19. tar 명령어

- 파일을 압축하고 압축을 해제하는 명령어

ex)

1. 압축할 때

tar cvf 압축파일명 압축파일대상

2. 압축을 해제할 때

tar xvf 압축파일명 압축해제할위치

c : compress, 여러개의 파일을 하나로 만든다.
v : view, 압축되는 과정을 보여준다.
f : file, 생성되는 파일명을 지정
x : extract, 묶여있는 파일을 푼다.
-C : 압축이 풀릴 위치를 지정

문제96. 집에서 emp.txt 를 복사해서 emp1.txt, emp2.txt, emp3.txt를 생성하시오.

```
# cp emp.txt emp1.txt
# cp emp.txt emp2.txt
# cp emp.txt emp3.txt
# ls -l emp*.txt
```

문제97. 현재 디렉토리에 emp로 시작하는 모든 text파일을 empall.tar 라는 이름으로 압축하시오.

```
# tar cvf empall.tar ./emp*.txt
# ls -l empall.tar
```

./ 는 현재 디렉토리를 표현

문제98. 현재 디렉토리에 test200이라는 디렉토리를 만들고 empall.tar 파일을 test200 디렉토리 밑에 복사하시오.

```
# mkdir test200
# cp empall.tar ./test200/
```

문제99. 현재 디렉토리에서 test200 디렉토리로 이동해서 empall.tar파일의 압축을 해제하시오.

```
# cd test200
# tar xvf empall.tar
# ls
```

문제100. /root 밑에 있는 dept.txt를 dept1.txt, dept2.txt, dept3.txt로 복사하고 /root 밑에 있는 dept로 시작하는 텍스트 파일들을 deptall.tar로

압축한 후에 /root 밑에 test500이라는 폴더를 만들고 deptall.tar를 test500 밑에 복사하고 압축을 푸시오.

```
# cp dept.txt dept1.txt
# cp dept.txt dept2.txt
# cp dept.txt dept3.txt
# ls -l dept*.txt
# tar cvf deptall.tar ./dept*.txt
# ls -l deptall.tar
# mkdir test500
# cp deptall.tar ./test500/
# cd test500
# tar xvf deptall.tar
# ls
```

문제101. 윈도우에서 emp.txt와 dept.txt를 압축 하시오.
(압축파일명은 fileall.tar)

문제102. fileall.tar 압축파일을 리눅스의 바탕화면에 가져다 놓으시오.

문제103. /root 밑에 있는 fileall.tar를 압축해제하시오.

문제104. 윈도우에서 jobs.txt를 jobs2.txt로 복사한 후에 두개의 파일을 jobs.zip으로 압축하고 리눅스 서버의 home 폴더에 넣으시오.

바탕화면 디렉토리를 터미널창에서 이동

```
# cd /root/Desktop
# pwd
# ls
```

20. ln 명령어

- os 윈도우의 바로가기 기능과 유사하다.

ex)

```
# cd
# ls -l find_job.sh
# cat find_job.sh

# ln -s /root/find_job.sh /root/Desktop/find_job.sh
```

/root밑의 find_job.sh를 바탕화면 (/root/Desktop) 에 find_job.sh 바로가기를 생성

21. sed 명령어

grep 명령어는 파일의 특정 내용을 검색하는 기능을 갖는다면 sed명령어는 검색뿐만 아니라 내용을 변경할 수도 있다.

ex)
cd
sed 's/KING/yyy/' emp.txt

emp.txt에서 KING을 yyy로 변경한다.
그러나 위의 명령어는 변경해서 보여주는 것이지 실제로 데이터가 변경되지는 않는다.

문제105. 위에서 KING이름을 yyy 로 변경해서 보여주고 있는 결과화면을 저장해서 emp900.txt 로 저장하시오.

sed 's/KING/yyy/' emp.txt >> emp900.txt

22. case 문

if 문과 유사한 리눅스의 쉘 스크립트 명령어

ex)
find_sal.sh 와 find_file.sh가 잘 수행되는지 확인하시오.

sh find_sal.sh

sh find_file.sh

ex)
아래의 스크립트를 작성하시오.

```
echo " Press number 1 for confirm employees sal Press number 2 for confirm to search file "

echo -n "Press number"
read choice
case $choice in
        1)
                sh /root/find_sal.sh ;;
        2)
                sh /root/find_file.sh ;;

esac
```

ex)
/root 밑에서 vi a.sh를 열어서 위의 스크립트를 복사해서 붙여넣고 아래와 같이 실행하시오.

cd
vi a.sh

sh a.sh

문제106. /root 밑에 있는 find_loc2.sh를 실행하시오.

```
# sh find_loc2.sh
```

문제107. find_loc2.sh를 위의 예제에서 만든 a.sh의 3번으로 추가하시오.

```
# vi a.sh

echo "Press number 1 for confirm employees sal
Press number 2 for confirm to search file
Press number 3 for confirm to find loc"

echo -n "Press number"
read choice
case $choice in
      1)
            sh /root/find_sal.sh ;;
      2)
            sh /root/find_file.sh ;;
      3)
            sh /root/find_loc2.sh ;;

esac

# sh a.sh
```

23. cp 문

- 파일을 복사하는 명령어

ex)
cp 파일명 복사할파일명
cp 위치/파일명 위치/복사할 파일명
cp emp.txt emp400.txt
cp /root/emp.txt /root/test01/emp400.txt

문제108. /root/ 밑에 있는 확장자가 .txt로 파일을 전부 /root/ 밑에 backup이라는 폴더를 만들고 모두 복사하시오.

```
# mkdir backup
# cp *.txt ./backup/
```

24. mv 명령어

- 파일의 이름을 바꾸거나 파일을 다른 디렉토리로 이동하는 명령어

ex)
mv 기존파일명 새로운파일명
mv emp.txt emp500.txt
mv emp500.txt ./backup/

emp500.txt를 현재 디렉토리 밑에 backup밑으로 이동

문제109. 집으로 와서 backup2폴더를 만들고 집에 있는 텍스트 파일들을 전부 backup2폴더로 이동시키시오.

```
# mkdir backup2
# mv *.txt ./backup2/
# ls *.txt
# cd backup2
# ls *.txt
```

리눅스 centos7에서 아나콘다 설치하기

아나콘다 설치하고 spyder 설치하는작업을 root에서 하지 않고 scott에서 수행해야 한다.

1. 리눅스에서 firefox를 실행하고 아래의 url로 접속한다.

https://velog.io/@shchoice/installpythonandanacondaincentos7

copy link location 을 눌러서 링크를 복사한다.

2. 터미널 창을 열고 wget 명령어로 설치파일을 내려받는다.

wget https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-x86_64.sh

3. 설치 파일을 실행한다.

```
# sh Anaconda3-2020.11-Linux-x86_64.sh
```

4. 리눅스의 환경설정 파일 bashrc 파일을 수행한다.

```
# source .bashrc
```

5. 아나콘다 가상환경 만들기

아나콘다 가상환경이란
머신러닝을 위한 파이썬 환경과 딥러닝을 위한 파이썬 환경이 서로 다르므로 각각 별도의 환경을 만들어서 파이썬을 사용하고 싶다면 가상환경을 생성하면 된다.

```
# conda create -n py38

# conda activate py38
```

6. spyder 설치

```
# conda install spyder
```

```
# pip install PyQt5==5.10

# pip install pyqtwebengine==5.12

# pip install pyqt5==5.12
```

7. spyder 실행

```
# spyder
```

문제110. 주사위를 10000번 던져서 주사위의 눈이 3이 나올 확률을 출력하시오.

```
# vi a.py

import random
dice = [1, 2, 3, 4, 5, 6]
cnt = 0

for i in range(10000):
    result = random.choice(dice)
    if result == 3:
        cnt += 1

print (cnt/10000)

# python a.py
```

21.01.04

2021년 1월 4일 월요일 오전 10:02

vi편집기 명령어

1. command 모드
- vi 편집기의 기본 모드이며 vi를 실행하면 바로 보이는 화면 (방향키로 왔다갔다 할 수 있다.)
2. edit 모드
- a, i, o, x 등을 누르면서 내용을 입력 또는 삭제하는 명령 모드
3. last line 모드
- 입력모드에서 저장, 종료, 강제종료 등의 령어를 입력하는 모드

    :wq - 저장 후 종료 (ZZ)

    :q - 저장 하지 않고 종료 (ZQ)

vi 편집기내에서 커서이동

1. j : 아래로 이동
2. k : 위로 이동
3. h : 왼쪽으로 이동
4. l : 오른쪽으로 이동
5. 1G : 맨위로 이동
6. G : 맨 아래로 이동
7. :set nu : 파일내의 텍스트에 번호표시
8. :set nonu : 번호를 안보이게 하는 명령어
9. gg : 맨위로 이동하는 단축키

vi 편집기의 삭제 명령어

1. x : 철자 하나를 삭제
2. dd : 한 행 삭제
3. dw : 커서가 있는 단어 삭제
4. :5,10 d : 5~10 번째 행 삭제
5. D : 커서 오른쪽 행 삭제

vi 편집기의 취소 명령어

1. u : 방금 작업한 내용을 취소

vi 편집기의 수정 명령어

1. a : 커서 다음에 입력
2. i : 커서 전에 입력
3. r : 커서에 위치하는 철자를 수정

문제111. 동전을 10만번 던져서 앞면이 나오는 횟수를 파이썬으로 출력하시오.

```
# vi coin_cnt.py

import random
coin = ['front','back']
cnt = 0

for i in range(100000):
	result = random.choice(coin)
	if result == 'front':
		cnt += 1

print (cnt/100000)

# python coin_cnt.py
```

vi 편집기의 복사/붙여넣기 명령어

1. yy : 하나의 행을 복사
2. p : 붙여넣기
3. yG : 현재행 부터 파일 끝까지 복사
4. :1,2 co 3 : 1~2행을 3행 다음으로 복사
5. :1,2 m 3 : 1~2행을 3행 다음으로 이동

문제112. jobs.txt를 열어서 맨 위에 한줄을 복사해서 다음라인데 붙여넣으시오.

```
$ vi jobs.txt
1G : 맨 위로 이동
yy : 한 행 복사
p : 붙여넣기
```

문제113. jobs.txt를 열어서 전체를 복사한 후에 맨 밑에 전체를 붙여 넣으시오.

```
$ vi jobs.txt
```
yG : 전체 복사

G : 맨 아래로 이동

p : 붙여넣기

dG : 커서위치부터 맨 아래까지 삭제

문제114. jobs.txt를 열어서 위아래 좌우로 움직이시오.

```
$ vi jobs.txt
```
j : 아래

k : 위

h : 왼쪽

l : 오른쪽

문제115. jobs.txt를 열어서 라인마다 숫자번호가 붙게 하시오.

```
$ vi jobs.txt
:set nu
```

문제116. jobs.txt에서 1번 ~ 5번째 라인까지 복사해서 jobs2.txt를 생성하시오.

```
$ vi jobs.txt
:1,5 w jobs2.txt
$ cat jobs2.txt
```

문제117. jobs.txt를 열어서 1번 ~ 20번 라인까지의 내용을 jobs7.txt로 생성하시오.

```
$ vi jobs.txt
:1,20 w jobs7.txt
$ wc -l jobs7.txt
```

문제118. 구구단 2단을 출력하는 파이썬 코드를 리눅승서 직접 작성하고 아래와 같이 실행되게 하시오.

```
$ vi p_2_dan.py

for i in range (1, 10):
    print (2, 'x', i, '=', 2*i)
```

```
$ python p_2_dan.py
```

문제119. 구구단 전체를 출력하시오.

```
$ vi gugu_dan.py

for i in range(2, 10):
    for j in range(1, 10):
        print (i, 'x', j, '=', i*j)

$ python gugu_dan.py
```

vi 편집기 내에서 특정문자를 검색하는 방법

:/검색할 문자

ex)
$ vi jobs.txt

:/about

n을 누르면 전진하면서 다음 about을 검색
shift + n을 누르면 후진하면서 이전 about을 검색

문제120. emp.txt를 열어서 SCOTT이라는 문자가 있는지 검색하시오.

```
$ vi emp.txt

:/SCOTT
```

vi 편집기 명령어로 문자를 변경하는 방법

:%s/기존문자/변경할문자/g

ex)
emp.txt 에서 KING을 aaa로 변경하시오.
$ vi emp.txt

:%s/KING/aaa/g

문제121. aaa를 다시 KING로 변경하시오.

```
$ vi emp.txt
```

:%s/aaa/KING/g

모든 데이터를 변경하는게 아니라 하나만 변경할 때

:s/기존문자/변경할문자

지금커서가 있는 현재행의 기존문자를 변경할 문자로 변경

ex)
$ vi emp.txt

:s/SALESMAN/aaa

커서가 있는 하나의 SALESMAN만 aaa로 변경된다.

문제122. emp.txt에서 이름이 ALLEN인 사원의 직업을 bbb로 변경하시오.

$ vi emp.txt

:s/SALESMAN/bbb

문제123. emp.txt에서 직업이 MANAGER인 사원들의 직업을 모두 SALESMAN으로 변경하시오.

$ vi emp.txt

:%s/MANAGER/SALESMAN/g

문제124. emp.txt를 emp1.txt ~ emp20.txt로 복사하시오.

```
$ cp emp.txt emp1.txt
$ cp emp.txt emp2.txt
$ cp emp.txt emp3.txt
                :
                :
$ cp emp.txt emp20.txt
```

문제125. emp1.txt ~ emp20.txt 의 내용중에 SALESMAN을 jjj로 변경하시오.

$ vi emp*.txt

```
:argdo %s/SALESMAN/jjj/g | update
```

argdo를 이용하면 여러개의 파일을 한번에 변경할 수 있다.

문제126. emp1.txt ~ emp20.txt 의 내용중에 jjj를 SALESMAN으로 변경하시오.

```
$ vi emp*.txt

:argdo %s/jjj/SALESMAN/g | update
```

문제127. jobs.txt를 jobs1.txt ~ jobs10.txt로 복사하시오.

```
$ cp jobs.txt jobs1.txt
$ cp jobs.txt jobs2.txt
$ cp jobs.txt jobs3.txt
                :
                :
$ cp jobs.txt jobs10.txt
```

문제128. jobs1.txt ~ jobs10.txt를 열어서 about를 aaa로 변경하시오.

```
$ vi jobs*.txt

:argdo %s/about/aaa/g | update
```

권한 관리 명령어

리눅스에 하둡을 설치하고 운영을 할 때 여러가지 문제들이 발생하는데 그중의 많은 문제들이 권한에 관련한 오류들이 가장 많다.
그래서 하둡운영을 원활하게 하기 위해서는 권한관리 명령어를 잘 숙지하고 있어야 한다.

권한 관리 명령어 3가지

1. chmod : change mode
2. chown : change ownership of a file
3. chattr : change file attributes

권한 관리 표

번호	권한	대표문자	파일	디렉토리

4	읽기권한	r	읽기, 복사	디렉토리에서 ls 가능
2	쓰기권한	w	수정	디렉토리에서 파일생성 가능
1	실행권한	x	실행	디렉토리에서 cd로 이동가능

ls -l로 어떤 특정파일을 조회했을 때 나오는 권한 부분 해석

```
$ ls -l emp.txt
-rwxrwxrwx. 1 scott scott 1029 Jan  4 01:09 emp.txt
```

위에서 -rwxrwxrwx. 부분이 권한 부분이다.

- r w x r w x r w x.
- : 파일이면 -, 디렉토리면 d 로 출력된다.

r w x : 파일의 소유자 권한
r w x : 파일의 그룹에 속한 유저들의 권한
r w x. : 기타유저들의 권한

문제129. 숫자 1부터 10까지 출력하는데 a20.py라는 파이썬 파일을 생성하시오.

```
$ vi a20.py

for i in range(1,11):
    print (i)

$ python a20.py
```

```
$ ls -l a20.py
-rw-rw-r--. 1 scott scott 33 Jan  4 01:54 a20.py
```

소유자는 읽고(r) 쓰는(w) 권한을 가지고 있고 실행권한은없다.
그룹유저들도 읽고(r) 쓰는(w) 권한을 가지고 있고 실행권한은없다.
기타유저들은 읽을(r) 수 있는 권한만 있다.

문제130. a20.py 소유자가 실행할 수 있는 권한을 가질 수 있도록 하시오.

```
$ chmod u+x a20.py

$ ls -l a20.py
-rwxrw-r--. 1 scott scott 33 Jan  4 01:54 a20.py
```

u+x : 소유자에게 실행권한을 준다.

u-x : 소유자가 가지고있는 실행권한을 뺏는다.
u+r : 소유자에게 읽기권한을 준다.
u-r : 소유자가 가지고있는 읽기권한을 뺏는다.
u+w : 소유자에게 쓰기권한을 준다.
u-w : 소유자가 가지고있는 쓰기권한을 뺏는다.
u+rwx : 소유자에게 읽기, 쓰기, 실행권한을 준다.

문제131. a20.py의 소유자가 a20.py를 읽고, 쓰고, 실행할 수 있는 권한을 모두 뺏으시오.

```
$ chmod u-rwx a20.py

$ ls -l a20.py
----rw-r--. 1 scott scott 33 Jan  4 01:54 a20.py
```

문제132. a20.py 소유자가 a20.py를 읽고, 쓸수 있도록 권한을 주시오.

```
$ chmod u+rw a20.py

$ ls -l a20.py
-rw-rw-r--. 1 scott scott 33 Jan  4 01:54 a20.py
```

문제133. a20.py의 소유자는 a20.py를 읽고, 쓰고 실행할 수 있게 하고 그룹유저들과 기타 유저들은 아무것도 못하게 하시오.

```
$ chmod u+rwx,g-rwx,o-rwx a20.py

$ ls -l a20.py
-rwx------. 1 scott scott 33 Jan  4 01:54 a20.py
```

root 유저는 scott와는 다르게 모든 파일에 대해서 읽고, 쓰고, 실행할 수 있는 권한을 가질 수 있도록 chmod 명령어를 수행할 수 있다.

문제134. root유저로 접속하시오.

```
$ su -

# whoami
```

문제135. 다시 scott으로 접속하시오.

```
# su - scott
```

```
$ whoami
```

문제136. root로 접속하고 scott의 집인 /home/scott 으로 이동하시오.

```
$ su -

# cd /home/scott
# pwd
```

root는 최고권한자 이기 때문에 a20.py를 볼 수도 있고 권한도 변경할 수 있다.

문제137. root가 a20.py의 소유자가 가지고 있는 읽고, 쓰고, 실행할 수 있는 권한을 모두 뺏으시오.

```
# chmod u-rwx a20.py

# ls -l a20.py
----------. 1 scott scott 33 Jan  4 01:54 a20.py
```

문제138. 다시 scott로 터미널 창에서 접속하고 scott에서 a20.py를 vi로 열어 보시오.

```
# su - scott

$ vi a20.py
```

pemission denied라는 메세지가 나오면서 열리지 않지만 소유자는 scott이므로 권한을 다시 직접 넣을 수 있다.

문제139. scott 유저에서 a20.py 에 대해 읽고, 쓰고, 실행할 수 있는 권한을 넣으시오.

```
$ chmod u+rwx a20.py

$ ls -l a20.py
-rwx------. 1 scott scott 33 Jan  4 01:54 a20.py
```

문제140. 리눅스에서 spyder를 실행해서 아래의 통계문제를 푸시오.

도박사의 동전 10번 던져서 뒷면이 나오는 결과를 아래와 같이 출력되게 하시오.

동전을 10번 던졌을 때 뒷면이 나오는 횟수가 0번이 나올 확률은 신뢰구간 95%안에 없습니다.
동전을 10번 던졌을 때 뒷면이 나오는 횟수가 1번이 나올 확률은 신뢰구간 95%안에 없습니다.
동전을 10번 던졌을 때 뒷면이 나오는 횟수가 2번이 나올 확률은 신뢰구간 95%안에 있습니다.
동전을 10번 던졌을 때 뒷면이 나오는 횟수가 3번이 나올 확률은 신뢰구간 95%안에 있습니다.
동전을 10번 던졌을 때 뒷면이 나오는 횟수가 4번이 나올 확률은 신뢰구간 95%안에 있습니다.
동전을 10번 던졌을 때 뒷면이 나오는 횟수가 5번이 나올 확률은 신뢰구간 95%안에 있습니다.
동전을 10번 던졌을 때 뒷면이 나오는 횟수가 6번이 나올 확률은 신뢰구간 95%안에 있습니다.
동전을 10번 던졌을 때 뒷면이 나오는 횟수가 7번이 나올 확률은 신뢰구간 95%안에 있습니다.
동전을 10번 던졌을 때 뒷면이 나오는 횟수가 8번이 나올 확률은 신뢰구간 95%안에 있습니다.
동전을 10번 던졌을 때 뒷면이 나오는 횟수가 9번이 나올 확률은 신뢰구간 95%안에 없습니다.
동전을 10번 던졌을 때 뒷면이 나오는 횟수가 10번이 나올 확률은 신뢰구간 95%안에 없습니다.

```
import random
import numpy as np

def coin_avg_std(num):
    result=[]
    for i in range(num):
        cnt = 0
        for j in range(10):
            cnt += (random.randint(0,1))
        result.append(cnt)
    c_m = np.mean(result)
    c_s = np.std(result)
    return c_m, c_s

def coin_hypo(num):
    c_m, c_s = coin_avg_std(10000)
    if c_m - 1.96 * c_s <= num <= c_m +1.96 * c_s:
        return f'동전을 10번 던졌을 때 뒷면이 나오는 횟수가 {num}번이 나올 확률은 신뢰구간 95%안에 있습니다.'
    else:
        return f'동전을 10번 던졌을 때 뒷면이 나오는 횟수가 {num}번이 나올 확률은 신뢰구간 95%안에 없습니다.'

for k in range(11):
    print(coin_hypo(k))
```

21.01.05

2021년 1월 5일 화요일 오전 9:40

리눅스 권한관리 명령어

1. chmod : 특정파일의 권한을 조정하는 명령어
2. chown : 특정파일이나 디렉토리의 소유자를 변경하는 명령어
3. chattr : 루트 유저만 권한을 조정할 수 있도록 설정하는 명령어

문제141. dept.txt의 권한이 어떻게 되어 있는지 확인하시오.

```
$ ls -l dept.txt
-rw-r--r--. 1 root root 82 Jan  5 09:47 dept.txt
```

문제142. dept.txt에서 권한이 아래와 같이 나타나게 하시오.

-rwxr-xr-x. 1 scott scott 82 Jan 5 10:03 dept.txt

```
$ chmod g-w,o-w dept.txt

$ ls -l dept.txt
-rwxr-xr-x. 1 scott scott 644 Jan  5 10:03 dept.txt
```

chown 명령어

- 파일이나 디렉토리의 소유자를 변경하는 명령어

ex) 소유자를 확인하는 방법

```
$ ls -l dept.txt
-rwxrwxrwx. 1 root root 644 Dec 29 14:15 emp.txt
```

root로 접속해서 emp.txt의 소유자를 scott으로 변경한다.

```
# chown scott:scott emp.txt
# ls -l emp.txt
-rwxrwxrwx. 1 scott scott 644 Jan  5 10:03 emp.txt
```

문제143. dept.txt의 소유자를 scott으로 변경하고 그룹도 scott으로 변경하시오.

```
# chown scott:scott dept.txt

# ls -l dept.txt
```

-rwxrwxrwx. 1 scott scott 82 Jan 5 10:03 dept.txt

문제144. scott 으로 switch user 하시오.

su - scott

문제145. emp.txt의 권한을 아래와 같이 나타나게 하시오.

-rwxr-xr-x. 1 scott scott 644 Jan 5 10:03 emp.txt

$ chmod g-w,o-w emp.txt

$ ls -l emp.txt
-rwxr-xr-x. 1 scott scott 644 Jan 5 10:03 emp.txt

문제146. dept.txt도 emp.txt와 같이 권한이 관리되게 하시오.

$ chmod g-w,o-w dept.txt

$ ls -l dept.txt
-rwxr-xr-x. 1 scott scott 644 Jan 5 10:03 dept.txt

문제147. dept.txt 의 권한이 아래와 같이 나타나게 하시오.

-rwx------. 1 scott scott 82 Jan 5 10:03 dept.txt

$ chmod g-rwx,o-rwx dept.txt

$ ls -l dept.txt
-rwx------. 1 scott scott 82 Jan 5 10:03 dept.txt

숫자로 권한을 변경하는 방법

번호	권한	대표문자	파일	디렉토리
4	읽기권한	r	읽기, 복사	디렉토리에서 ls 가능
2	쓰기권한	w	수정	디렉토리에서 파일생성 가능
1	실행권한	x	실행	디렉토리에서 cd로 이동가능

ex)
emp.txt의 권한을 유저, 그룹, 기타 유저 모두 읽을 수만 있도록 변경하시오.

$ chmod u-rwx,g-rwx,o-rwx emp.txt
$ chmod u+r,g+r,o+r emp.txt

위의 작업을 숫자로 하시오.

$ chmod u-rwx,g-rwx,o-rwx emp.txt
$ chmod 444 emp.txt
$ ls -l emp.txt
-r--r--r--. 1 scott scott 644 Jan 5 10:03 emp.txt

문제148. emp.txt에 대해서 아래와 같이 권한이 들어가게 하시오.

-rw-rw-rw-. 1 scott scott 644 Jan 5 10:03 emp.txt

$ chmod 666 emp.txt

$ ls -l emp.txt
-rw-rw-rw-. 1 scott scott 644 Jan 5 10:03 emp.txt

문제149. emp.txt에 대해서 아래와 같이 권한이 들어가게 하시오.

-rwxrwxrwx. 1 scott scott 644 Jan 5 10:03 emp.txt

$ chmod 777 emp.txt

$ ls -l emp.txt
-rwxrwxrwx. 1 scott scott 644 Jan 5 10:03 emp.txt

문제150. emp.txt에 대해서 아래와 같이 권한이 들어가게 하시오.

---x--x--x. 1 scott scott 644 Jan 5 10:03 emp.txt

$ chmod 111 emp.txt

$ ls -l emp.txt
---x--x--x. 1 scott scott 644 Jan 5 10:03 emp.txt

문제151. emp.txt에 대해서 아래와 같이 권한이 들어가게 하시오.

-rwxr-x--x. 1 scott scott 644 Jan 5 10:03 emp.txt

$ chmod 751 emp.txt

$ ls -l emp.txt
-rwxr-x--x. 1 scott scott 644 Jan 5 10:03 emp.txt

문제152. emp.txt에 대해서 아래와 같이 권한이 들어가게 하시오.

----------. 1 scott scott 644 Jan 5 10:03 emp.txt

$ chmod 000 emp.txt

$ ls -l emp.txt
----------. 1 scott scott 644 Jan 5 10:03 emp.txt

문제153. emp.txt에 대해서 아래와 같이 권한이 들어가게 하시오.

-rwxrw-rw-. 1 scott scott 644 Jan 5 10:03 emp.txt

$ chmod 766 emp.txt

$ ls -l emp.txt
-rwxrw-rw-. 1 scott scott 644 Jan 5 10:03 emp.txt

디렉토리의 소유자의 권한을 변경하는 방법

디렉토리의 소유자의 권한을 변경하게 되면 그 디렉토리에 있는 모든 파일들의 소유자를 한번에 변경할 수 있다.

ex1)
scott 유저에서 /home/scott 밑에 test500 디렉토리를 생성

$ mkdir test500

ex2)
test500 디렉토리 밑에 emp.txt와 dept.txt를 복사한다.

$ cp *.txt ./test500/
$ ls -l ./test500/

ex3)
test500 디렉토리의 소유자가 누구인지 확인하시오.

$ ls -ld test500
drwxrwxr-x. 2 scott scott 37 Jan 5 11:11 test500

ex4)
test500 디렉토리의 소유자를 root로 변경하시오.

$ su -

cd /home/scott

ls -ld test500
drwxrwxr-x. 2 scott scott 37 Jan 5 11:11 test500

chown root:root test500

ls -ld test500
drwxrwxr-x. 2 root root 37 Jan 5 11:11 test500

cd test500

ls -l *.txt
-rwx------. 1 scott scott 82 Jan 5 11:11 dept.txt

-rwxrw-r--. 1 scott scott 644 Jan 5 11:11 emp.txt

test500 디렉토리의 소유자는 root로 변경 되었지만 test500 디렉토리 안의 emp.txt와 dept.txt 의 소유자는 scott로 유지 된다.

ex5)
test500 디렉토리의 소유자를 root로 변경하면서 test500 디렉토리 안에 있는 텍스트 파일의 소유자도 root로 변경하시오.

cd /home/scott

chown -R root:root test500

ls -l ./test500/
total 8
-rwx------. 1 root root 82 Jan 5 11:11 dept.txt
-rwxrw-r--. 1 root root 644 Jan 5 11:11 emp.txt

-R 옵션을 붙이면 디렉토리의 소유자 뿐만 아니라 디렉토리 아래에 있는 모든 파일들과 하위 디렉토리의 소유자도 한번에 변경할 수 있다.

문제154. /home/scott 밑에 있는 test500 디렉토리와 그 밑에 있는 파일들의 소유자를 모두 scott로 변경하시오.

cd /home/scott

chown -R scott:scott test500

ls -ld test500
drwxrwxr-x. 2 scott scott 37 Jan 5 11:11 test500

]# ls -l ./test500/
total 8
-rwx------. 1 scott scott 82 Jan 5 11:11 dept.txt
-rwxrw-r--. 1 scott scott 644 Jan 5 11:11 emp.txt

chattr 명령어

- chattr 명령어를 이용하면 chmod 명령어 수행을 막을 수 있다.
  chattr 명령어는 최상위 계정인 root 에서만 수행할 수 있다.

ex)
su -
cd /home/scott
chattr +i emp.txt
lsattr emp.txt
----i----------- emp.txt

emp.txt에 대해서 root 유저 외 다른 유저는 삭제, 변경, 내용추가가 불가능하게 된다.

```
# su - scott
$ ls -l emp.txt
-rwxrw-rw-. 1 scott scott 644 Jan  5 10:03 emp.txt

$ chmod 777 emp.txt
chmod: changing permissions of ‘emp.txt’: Operation not permitted
```

소유자가 자기 자신인 scott 임에도 불구하고 chmod 명령어가 수행되지 않는다.

ex)
다시 scott이 emp.txt를 chmod 할 수 있도록 설정하시오.

```
$ su -
# cd /home/scott
# chattr -i emp.txt
# lsattr emp.txt
---------------- emp.txt

# su - scott
$ chmod 777 emp.txt

$ ls -l emp.txt
-rwxrwxrwx. 1 scott scott 644 Jan  5 10:03 emp.txt
```

chattr 명령어는 중요한 파일을 root만 변경할 수 있고 나머지 유저들은 읽기만 할 수 있도록 하고 싶을때 유용하다.

문제155. emp.txt에 대해서 아래와 같이 권한이 유지되게 하고 root 유저 외에는 chmod를 할 수 없도록 막으시오.

```
-r--r--r--. 1 scott scott 644 Jan  5 10:03 emp.tx

$ chmod 444 emp.txt
$ ls -l emp.txt
-r--r--r--. 1 scott scott 644 Jan  5 10:03 emp.txt

$ su -
# cd /home/scott
# chattr +i emp.txt
# lsattr emp.txt
----i----------- emp.txt

# su - scott
$ chmod 777 emp.txt
chmod: changing permissions of ‘emp.txt’: Operation not permitted
```

문제156. 다시 emp.txt를 scott 유저가 chmod 할 수 있도록 변경하시

오.

```
$ su -
# cd /home/scott

# chattr -i emp.txt
# lsattr emp.txt
---------------- emp.txt
```

문제157. dept.txt에 대해서 chmod 할 수 없도록 막고 dept.txt의 권한을 아래와 같이 유지되게 하시오.

```
-rw-r--r--. 1 scott scott 82 Jan  5 10:03 dept.txt
```

```
$ chmod 644 dept.txt
$ ls -l dept.txt
-rw-r--r--. 1 scott scott 82 Jan  5 10:03 dept.txt

$ su -
# cd /home/scott

# chattr +i dept.txt
# lsattr dept.txt
----i----------- dept.txt
```

문제158. 숫자 1부터 20까지 출력하는 파이썬 코드를 작성하고 그 파이썬 코드의 권한이 아래와 같이 들어가게 하시오.

```
$ ls -l a158.py
-rwxrw-rw-. 1 scott scott 33 Jan  5 12:13 a158.py

$ vi a158.py

for i in range(1,21):
        print (i)

$ python a158.py

$ chmod 766 a158.py
$ ls -l a158.py
-rwxrw-rw-. 1 scott scott 33 Jan  5 12:13 a158.py
```

리눅스 디스크 관리 명령어

디스크 관리 명령어 3가지

1. df 명령어
   - 현재 파일 시스템의 총 사용율을 확인하는 명령어

ex)
$ df -h

2. du 명령어
- 현재 파일/디렉토리의 디스크 사용량을 표시하는 명령어

ex)
$ du *.txt

$ du -c *.txt

합계를 확인하려면 -c 옵션을 붙인다.

$ du -ch *.txt

문제 159. /home/scott 밑에 있는 확장자가 .sh로 끝나는 파일들의 총 크기를 확인하시오.

$ du -c *.sh

3. sar 명령어
- 리눅스 서버가 너무 느려서 작업이 진행이 잘 되지 않을 때 디스크의 사용율을 확인하는 명령어

ex)
$ sar 1 100

%user 부분을 집중적으로 모니터링 하면 된다. 리눅스 서버에 부하가 심한 작업을 수행하면 %user 부분의 사용율이 100에 가깝게 올라간다.

부하를 주는 작업들
  - 악성 SQL을 수행
  - 무한루프를 수행

문제160. 아래와 같이 1~10000000000 을 출력하는 파이썬 프로그램을 돌리고 다른 터미널 창에서 sar로 모니터링 하시오.

$ vi a.py

```
for i in range(1,10000000000):
    print (i)
```

$ python a.py

$ sar 1 100

sar 명령어 결과의 컬럼 소개

%user : scott 유저와 같이 일반 유저가 사용하는 disk i/o
- 악성 SQL을 수행하거나 무한루프를 돌리면 %user 사용율이 올라간다.

%nice : cpu를 양보하는 친절도
%system : system 이 사용하는 disk i/o
%iowait : i/o를 일으키면서 얼마나 대기하는지 확인
%steal : 다른 프로세서의 자원을 얼마나 뺏고 있는지 확인
%idle : 작업을 안하고 있는 idle 한 상태

문제161. 리눅스 쉘 프로그램을 만들고 돌린 후 직접 확인할 수 없는 상황에서 확인하는 방법

sar 명령어로 수행한 결과를 text 파일로 생성되게 한다.

```
$ sar 1 20 >> sar_20210105.txt
$ vi sar_20210105.txt
```

문제162. 위의 명령어를 백그라운드로 돌아가게 하시오.

```
$ sar 1 20 >> sar_20210106.txt &
```

리눅스 프로세서 관리 명령어

1. ps 명령어
2. top 명령어
3. kill 명령어
4. jobs 명령어

jobs 명령어
- 동작중인 작업의 상태를 확인하는 명령어

상태정보 4가지

1. running : 실행중
2. stopped : 일시중단중
3. Done : 종료
4. terminated : 강제종료

ex)
$ vi hhh.txt

```
select ename, sal, job, deptno
        from emp
        where
```

esc 을 누르고 ctl +z (작업 취소)

```
[2]+  Stopped                 vim hhh.txt
```

$ jobs : 동작중인 작업의 상태를 확인하는 명령어
$ fg : 현재 진행중이었던 job으로 접속하는 명령어

문제163. 아래와 같이 일시중단된 job들을 여러 개 만드시오.

```
$ vi hhh2.txt

select
```

esc 키 누르고 ctl + z

```
$ vi hhh3.txt

select ename
```

esc 키 누르고 ctl + z

```
$ vi hhh4.txt

select ename, sal
```

esc 키 누르고 ctl + z

```
$ jobs
[1]   Stopped                 vim hhh2.txt
[2]-  Stopped                 vim hhh3.txt
[3]+  Stopped                 vim hhh4.txt
```

+ : 현재 job
- : 현재 이전에 현재 job 이었던 job

문제164. 위의 상태에서 2번 job에 들어가시오.

```
$ fg 2
```

문제165. 위의 상태에서 아래의 그라운드로 돌리는 sar명령어를 수행하고 출력되는 결과를 해석하시오.

```
$ sar 1 100 >> sar _2021.txt &
```

[4] 15022

[4] : job 번호
15022 : 프로세서 번호

ps 명령어

- 현재 시스템에서 수행되고 있는 프로세서의 정보를 표시하는 명령어

$ ps 옵션 프로세서번호
ex)
$ ps -p 15022

-p : 프로세서 아이디
-e : 현재 실행중인 모든 프로세서
-f : 실제 유저명, 개시시간등을 표시
-l : 프로세서의 상태, 우선도 등과 같은 상세한 정보를 표시

문제166. 현재 리눅스 시스템에 떠있는 모든 프로세서를 다 출력하시오.

```
$ ps -ef | grep scott
```

문제167. 현재 리눅스 시스템에 떠있는 프로세서들 중에서 vim으로 vi 편집기 작업중이었던 프로세서들만 조회하시오.

```
$ ps -ef | grep vim
```

kill 명령어

- 프로세서를 죽이는 명령어

문제168. 위의 vim으로 작업중이던 세션을 모두 죽이시오.

```
$ ps -ef | grep vim

scott    14903 14112  0 14:56 pts/0    00:00:00 vim hhh2.txt
scott    14911 14112  0 14:56 pts/0    00:00:00 vim hhh4.txt
scott    15167 14112  0 15:17 pts/0    00:00:00 grep --color=auto vim

$ kill -9 14903
$ kill -9 14911
$ kill -9 15167
```

$ jobs

-9 옵션을 사용하면 강제로 죽인다.

top 명령어

- 지금 현재 작동중인 프로세서들의 cpu 사용율과 메모리 사용율을 확인하는 명령어

내가 돌린 파이썬 프로그램이 cpu를 얼마나 많이 사용하고 있는지 확인하고 싶을때나 잘못 프로그래밍해서 무한루프를 돌렸거나 리눅스의 자원을 많이 사용하고 있다면 문제가 발생하는 것이므로 top 명령어로 확인해 볼 필요가 있다.

$ top

top에서 집중적으로 봐야할 부분은 cpu 사용율이 높은 프로세서가 무엇이고 어떤 작업을 하고 있는지 알아내야 한다.

문제169. 숫자 1~10000000000를 출력하는 파이썬 프로그램을 실행하고 다른 터미널 창을 열어서 top을 수행하시오.

$ python a.py

$ top

문제170. cpu를 많이 사용하고 있는 프로세서 15259 프로세서를 강제 종료 시키시오.

$ kill -9 15259

쉘스크립트 작성법

- 쉘(shell)이란 운영체제에서 제공하는 명령을 실행하는 프로그램이다.

- 쉘(shell) 스크립트란 인터프리터(통역사) 역활을 하는것으로 시스템에서 지원하는 명령어들의 집합을 묶어서 프로그램화 한것을 말한다.

- 쉘(shell)의 종류
  - Boune shell
  - C shell
  - Korn shell
  - bash shell : 가장 많이 사용

#!/bin/bash : 쉘 중에 bash 쉘을 사용하겠다는 뜻

쉘 프로그래밍 작성

- 쉘 스크립트 프로그래밍
  - c언어와 유사한 프로그래밍
  - 변수, 반복문(loop문), 제어문(if문)이 사용가능
  - 별도로 컴파일을 하지 않고 텍스트 파일 형태로 바로 실행이 가능하다.
  - vi로 작성이 가능
  - 리눅스의 많은 부분이 쉘 스크립트로 작성되어있다.

- 쉘 스크립트를 작성하고 실행하는 방법1

```
$ vi a.sh

echo "yahoo ~~"

$ sh a.sh
```

- 쉘 스크립트를 작성하고 실행하는 방법2

```
$ ./a.sh
-bash: ./a.sh: Permission denied
```

위와 같이 에러가 나오면 권한을 확인한다.

```
$ ls -l a.sh
-rw-rw-r--. 1 scott scott 16 Jan  5 16:09 a.sh
```

rw 권한은 있는데 실행권한이 없어서 에러가 발생한다.
소유자에게 실행권한을 넣어준다.

```
$ chmod 764 a.sh
$ ls -l a.sh
-rwxrw-r--. 1 scott scott 16 Jan  5 16:09 a.sh

$ ./a.sh
yahoo ~~
```

./파일명.sh 로 실행하려면 실행권한이 있어야한다.
sh 파일명.sh 는 파일의 실행권한이 없어도 실행된다.

- 변수 사용법

1. 모든 변수는 문자열(string)로 취급된다.
2. 변수 이름은 대소문자를 구분한다.
3. 변수에 값을 대입할 때는 '=' 좌우에 공백이 없어야 한다.
4. 변수에 들어간 문자를 출력하려면 변수 앞에 $를 붙이고 echo 명령어로 출력하면 된다.

ex)

```
$ myvar="Hi~~"
$ echo $myvar
Hi~~
```

- 변수의 숫자 계산하는 방법

1. 변수에 대입한 값은 두 문자열로 취급이 된다.
2. 변수에 들어있는 값을 숫자로 해서 사칙연산 (+, -, *, /)을 하려면 expr을 사용해야 한다.
3. 수식에 괄호 또는 곱하기(*)를 사용하려면 그 앞에 반드시 역슬래쉬(₩)를 붙여야 한다.

ex)

```
$ num1=100
$ num2=200

$ echo $num1
100

$ echo $num2
200

$ echo $num1 + $num2
100 + 200

$ expr $num1 + $num2
300
```

문제171. num1과 num2의 곱을 구하시오.

```
$ expr $num1 \* $num2
20000
```

곱하기(*)와 괄호 앞에는 역슬래쉬(₩)를 붙여주어야 한다.

문제172. 아래의 계산식을 구현하시오.

($num2 + 200) * $num1

```
$ expr \( $num2 + 200 \) \* $num1
40000
```

파라미터 변수

1. 파라미터 변수는 $0, $1, $2, ... 의 형태를 가진다.
2. 전체 파라미터는 $*로 표현한다.

ex)

```
$ vi add.sh

num1=$1
num2=$2
```

```
num3=`expr $num1 + $num2`
echo "$num1 와 $num2 를더하면 $num3 입니다."
```

```
$ sh add.sh 24 18
24 와 18 를더하면 42 입니다.
```

역따옴표(`)를 써야 역따옴표 안의 실행문이 실행이 되어서 num3변수에 할당 되어진다.

변수=`리눅스 명령어`

리눅스 명령어에 의해서 수행된 결과가 변수에 입력되어야 한다면 역따옴표(`)를 사용해야 한다.

문제173. 아래의 두수를 곱한 결과가 출력되는 쉘 스크립트를 작성하시오.

```
$ sh two_number.sh 12 54
12 와 54 를 곱하면 648 입니다.
```

```
$ vi two_number.sh
```

```
num1=$1
num2=$2
num3=`expr $num1 \* $num2`
echo "$num1 와 $num2 를 곱하면 $num3 입니다."
```

```
$ sh two_number.sh 12 54
12 와 54 를 곱하면 648 입니다.
```

21.01.06

2021년 1월 6일 수요일 오전 9:43

쉘스크립트 작성법

if 조건문에 들어가는 비교 연산자

- 문자열 비교
  - "문자열1"=="문자열2" : 두 문자열이 같으면 True
  - "문자열1"!="문자열2" : 두 문자열이 같지 않으면 True

- 숫자열 비교
  - 숫자1 -eq 숫자2 : 두 숫자가 같으면 True (=)
  - 숫자1 -ne 숫자2 : 두 숫자가 같지 않으면 True (!=)
  - 숫자1 -gt 숫자2 : 숫자1이 숫자2보다 크면 True (>)
  - 숫자1 -ge 숫자2 : 숫자1이 숫자2보다 크거나 같으면 True (>=)
  - 숫자1 -lt 숫자2 : 숫자1이 숫자2보다 작으면 True (<)
  - 숫자1 -le 숫자2 : 숫자1이 숫자2보다 작거나 같으면 True (<=)
  - !숫자1 : 숫자1이 거짓이라면 True (not)

ex)
$ vi if1.sh

```
if [ 100 -eq 200 ]; then
    echo "100 과 200 은 같습니다."
else
    echo "100 과 200 은 같지않습니다."
fi
```

```
$ sh if1.sh
100 과 200 은 같지않습니다.
```

문제174. 위의 스크립트에 파라미터 변수를 이용해서 아래와 같이 실행될 수 있게 하시오.

```
$ vi if1.sh

if [ $1 -eq $2 ]; then
    echo "$1 과 $2 은 같습니다."
else
    echo "$1 과 $2 은 같지않습니다."
fi
```

```
$ sh if1.sh 100 100
100 과 100 은 같습니다.
$ sh if1.sh 100 200
100 과 200 은 같지않습니다.
```

리눅스의 논리 연산자

1. and >> && 또는 -a
2. or >> || 또는 -o
3. not >> !

ex)

if [$sal -lt 2000] && [$job=="SALESMAN"]; then

또는

if [$sal -lt 2000 -a $ job=="SALESMAN"]; then

문제175. emp.txt에서 scott의 월급을 출력하시오.

```
$ awk '$2==toupper("scott") {print $6}' emp.txt
```

문제176. 위의 스크립트와 파라미터 변수를 이용해서 아래와 같이 실행되는 쉘 스크립트를 작성하시오.

$ sh find_sal.sh scott

scott 의 월급은 3000 입니다.

```
$ vi find_sal.sh

sal=`awk -v name=$1 '$2==toupper(name) {print $6}' emp.txt`
echo "$1 의 월급은 $sal 입니다."

$ sh find_sal.sh scott
```

문제177. 위의 스크립트를 이용해서 아래와 같이 실행되게 하시오.

$ sh find_job.sh allen

allen 의 직업은 SALESMAN 입니다.

```
$ vi find_job.sh
```

```
job=`awk -v name=$1 '$2==toupper(name) {print $3}' emp.txt`
echo "$1 의 직업은 $job 입니다."
```

```
$ sh find_job.sh allen
```

문제178. 위의 스크립트와 if문을 이용해서 아래의 쉘스크립트를 생성하시오. 월급을 2000 을 기준으로 월급의 인상여부를 출력하게 하시오.

```
$ sh find_sal.sh scott
```

```
월급 인상 대상자가 아닙니다.
```

```
$ sh find_sal.sh martin
```

```
월급 인상 대상자 입니다.
```

```
$ vi find_sal.sh
```

```
sal=`awk -v name=$1 '$2==toupper(name) {print $6}' emp.txt`
if [ $sal -lt 2000 ]; then
        echo "월급 인상 대상자 입니다."
else
        echo "월급 인상 대상자가 아닙니다."
fi
```

```
$ sh find_sal.sh scott
```

문제179. 위의 스크립트를 수정해서 부서번호가 30번이고 월급이 2000보다 작다면 '월급 인상 대상자 입니다.' 를 출력하고 그렇지 않다면 '월급 인상 대상자가 아닙니다.'를 출력하시오.

```
$ sh find_sal.sh martin
```

```
월급 인상 대상자 입니다.
```

```
$ vi find_sal.sh
```

```
sal=`awk -v name=$1 '$2==toupper(name) {print $6}' emp.txt`
deptno=`awk -v name=$1 '$2==toupper(name) {print $8}' emp.txt`

if [ ${sal:0:4} -lt 2000 -a ${deptno:0:2} -eq 30 ]; then
      echo "월급 인상 대상자 입니다."
else
      echo "월급 인상 대상자가 아닙니다."
fi
```

$ sh find_sal.sh martin

리눅스 쉘에서 loop문 사용법

1. for loop문 사용법

for 변수 in 값1,값2,값3
do
    반복할 문장
done

ex)
$ vi for1.sh

```
for i in {1..10}
do
    echo $i
done
```

$ sh for1.sh

문제180. 구구단 2단을 출력하시오.

$ vi 2_dan.sh

```
for i in {1..9} # 리눅스의 for loop문
do # loop문 시작
    let gop=2*$i # let 2*$i의 결과를 gop 변수에 할당
    echo "2 x $i = $gop"
done # loop문 끝
```

$ sh 2_dan.sh

이중 루프문

ex)
$ vi for2.sh

```
for i in {2..9}
do
    for k in {1..9}
    do
        echo "$i $k"
    done
done
```

$ sh for2.sh

문제181. 구구단 전체를 출력하시오.

$ vi for2.sh

```
for i in {2..9}
do
    for k in {1..9}
    do
        let gop=$i*$k
        echo "$i x $k = $gop "
    done
done
```

```
$ sh for2.sh
```

문제182. 아래의 별표를 출력하는 쉘스크립트를 작성하시오.

```
$ sh star.sh

★
★★
★★★
★★★★
★★★★★
```

```
$ vi star.sh

star="" # 더블 쿼테이션 마크 2개를 붙여 null값을 나타낸다.
for i in {1..5} # 1~5까지 loop
do
    star="$star★" # star = star*'★'
    echo $star # star 변수에 있는 값을 출력
done
```

```
$ sh star.sh
```

문제183. 위의 코드를 수정해서 숫자를 물어보게 하고 숫자를 입력하면 해당 숫자만큼 ★ 이 출력되게 하시오.

```
$ sh star.sh

숫자를 입력하시오. 5

★
★★
★★★
★★★★
★★★★★
```

```
$ vi star.sh

echo -n "숫자를 입력하시오."
read num
```

```
star=""
for i in `eval echo {1..$num}`
do
	star="$star★"
	echo $star
done
```

```
$ sh star.sh
```

for 반복문 안에서 echo {1..$num} 이 실행되게 하려면 eval을 사용해야 한다.

문제184. emp.txt를 복사해서 emp1.txt~emp100.txt로 복사하는 쉘스크립트를 작성하시오.

```
$ vi cp_emp.sh
```

```
for i in {1..100}
do
	cp emp.txt emp$i.txt
done
```

```
$ sh cp_emp.sh
```

문제185. 위에서 복사한 emp1.txt~emp100.txt의 이름을 employee1.txt~employee100.txt로 변경하시오.

```
$ vi mv_emp.sh
```

```
for i in {1..100}
do
	mv emp$i.txt employee$i.txt
done
```

```
$ sh mv_emp.sh
```

문제186. employee1.txt~employee100.txt 중 하나를 랜덤으로 골라서 파일을 연 후에 숫자 3000을 3900으로 변경하시오.

```
$ vi employee29.txt
```

```
:%s/3000/3900/g
```

문제187. emp.txt와 employee29.txt의 파일의 차이가 있는지 확인하시오.

```
$ diff --brief emp.txt employee29.txt
```

```
Files emp.txt and employee29.txt differ
```

문제188. for loop문을 이용해서 employee1.txt~employee100.txt 중 emp.txt와 차이가 있는 파일이 무엇인지 한번에 알아내시오.

```
$ vi diff_emp.sh

for i in {1..100}
do
    diff --brief emp.txt employee$i.txt
done

$ sh diff_emp.sh
Files emp.txt and employee29.txt differ
```

문제189. employee로 시작하는 text파일을 모두 지우시오.

```
$ rm employee*.txt
```

문제190. ls -l find_sal.sh 를 했을 때 출력되는 결과에서 크기에 해당하는 부분만 출력하시오.

```
$ ls -l find_sal.sh | awk '{print $5}'
```

문제191. size100 이라는 폴더를 만드시오.

```
$ mkdir size100
```

문제192. 확장자가 .sh 인 쉘스크립트 중에서 사이즈가 100바이트 이상인 사이즈만 출력하시오.

```
$ vi mv_size_100.sh

size=`ls -l *.sh | awk '{print $5}'` # .sh가 확장자인 파일의 사이즈 부분만 출력

for i in $size # 사이즈들을 하나씩 루프문으로 가져온다.
do
    if [ ${i:0:9} -gt 100 ];then # ${i:0:9} 문자형인 i를 9번째 자리까지 숫자형으로 변경
    echo $i
    fi
done

$ sh mv_size_100.sh
```

문제193. 확장자가 .sh 인 쉘스크립트 중에서 사이즈가 100바이트 이

상인 쉘 스크립트는 size100 폴더에 이동 시키시오.

```
$ vi mv_size_100.sh

list=`ls -l *.sh | awk '{print$9}'`

for i in $list
do
      size=`ls -l $i |awk '{print$5}'`
      if [ $size -ge 100 ];then
      mv $i size100
      fi
done

$ sh mv_size_100.sh
```

리눅스 쉘을 현업에서 활용하는 방법

1. 고객 데이터를 리눅스 쉘을 이용해서 데이터를 필터링 한다. : 리눅스 쉘
   - ex) 라이나 생명 고객중 40대만 따로 분리해서 text 파일을 생성한다.
2. 40대의 데이터를 가지고 군집분석(k-means)을 한다. : R 과 파이썬
3. 40대의 데이터를 가지고 연관분석(아프리오 알고르즘)을 한다. : R 과 파이썬

새로운 보험 상품이 출시되면 보험 가입이 가능할 것 같은 고객들만 필터링해서 연락을 하는게 훨씬 효과적이기 때문에 필터링한다.

문제194. emp.txt에서 직업이 SALESMAN인 사원들의 data만 가지고 salesman.txt라는 파일을 생성하시오.

```
$ awk '$3==toupper("salesman") {print $0}' emp.txt >> salesman.txt
```

문제195. emp.txt에서 직업을 출력하는데 중복을 제거해서 출력하시오.

```
$ awk '{print $3}' emp.txt | sort | uniq
```

문제196. 위의 직업들을 하나씩 불러와서 for loop문에서 출력하는 코드를 작성하시오.

```
$ vi for_job.sh

job=`awk '{print $3}' emp.txt | sort | uniq`

for i in $job
do
      echo $i
```

```
done

$ sh for_job.sh
```

문제197. 위의 스크립트를 이용해서 해당 직업의 사원들의 데이터만 직업을 이름으로 해서 아래와 같이 생성되게 하시오.

```
$ sh generate_job.sh

salesman.txt    manager.txt    …

$ vi generate_job.sh

job=`awk '{print $3}' emp.txt | sort | uniq`

for i in $job
do
    grep -i $i emp.txt >> $i.txt
    cat $i.txt
done

$ sh generate_job.sh
```

문제198. 위의 스크립트를 이용해서 부서번호별 emp.txt가 나눠지게 스크립트를 생성하시오.

```
$ sh make_deptno.sh

$ vi make_deptno.sh

deptno=`awk '{print $8}' emp.txt | sort | uniq`

for i in $deptno
do
    awk -v num=$i '$8==num {print $0}' emp.txt >> deptno${i:0:2}.txt
done

$ sh make_deptno.sh
```