1주차

▼ index

리눅스 쉘 사용법

리다이렉션 / 파이프

스트림

: command로 실행되는 process는 세 가지 스트림을 갖고 있음

- standard input stream
- standard output stream
- standard error stream

리다이렉션

: 스트림의 흐름을 바꿔주는 것

: `>` 또는 `<` 사용

ex)

1. ls `>` files.txt
   - ls로 출력되는 standard output stream의 방향을 files.txt로 바꿔줌
   - files.txt에 ls로 출력되는 결과가 저장
2. head `<` files.txt
   - 명령어의 standard input stream을 파일로 설정
   - files.txt의 파일 내용이 head라는 파일의 처음부터 10라인(head 명령어 default)까지 출력해주는 명령
3. head `<` files.txt `>` files2.txt
   - 앞은 동일
   - head의 output stream이 files2.txt로 저장됨
4. ls `>>` files.txt
   - 기존 파일에 추가됨
   - files.txt 파일의 끝에 ls의 출력 결과 추가

파이프

: 두 프로세스 사이에서 한 프로세스의 출력 스트림이 또 다른 프로세스의 입력 스트림으로 사용될 때 쓰임

1. ls `|` grep files.txt
    - ls로 출력되는 내용이 grep 명령의 입력 스트림으로 들어감
    - grep은 files.txt라는 문자열이 들어있는 입력 내용만 출력해주므로 결과는 현재 디렉토리 내 files.txt의 존재 여부를 알려줌
    - ls > grep files.txt는 왜 안 될까?
        - grep은 입력 스트림이 필요함. 리다이렉션을 사용하면 스트림의 형태를 바꿀 수 없음.
    - “또는"의 | 와 구분?
        - “또는"의 의미로 사용될 때는 quote 안에서 사용된다.
            - ex) grep -E “go|java” files.txt

grep

: standard input 또는 File 내에서의 검색 명령

```
grep [-option] [pattern] [file or directory name] ...
```

참고) `man [명령]`을 이용하면 명령에 대한 자세한 설명을 볼 수 있음

find

: 파일, 디렉토리, 링크 등에 대한 검색 명령

```
find [path] [-option] ...
```

프로세스 관리

프로세스

: 실행 중인 프로그램

- 프로그램의 실행/스케줄링 단위로 unique한 pid(process id)를 부여받음

Foregroud vs Background

- Foreground process: 쉘에서 해당 프로세스 실행을 명령한 후, 해당 프로세스의 수행 종료까지 다른 입력을 할 수 없는 프로세스
  - default로 수행됨
  - 프로세스 중지 `[Ctrl] + Z`
    - suspend 상태로 변경
    - 직후 `bg` 명령으로 최근에 중지된 프로세스를 백그라운드에서 실행시킬 수 있음
  - 프로세스 종료 `[Ctrl] + C`
- Background process: 사용자 입력과 상관없이 실행되는 프로세스
  - 맨 뒤에 `&`를 붙여주면 됨.
  - ex) PC에 있는 모든 파이썬 파일을 찾아서 list.txt에 저장하라(백그라운드로)

    ```
    find / -name "*.py" > list.txt &
    ```

  - 백그라운드로 실행하면 `[1] 57` 같은 출력이 나오는데, `[1]`은 **작업 번호**, `57`은 **pid**

프로세스 상태 확인

```
ps [option(s)]
```

프로세스 중지

```
kill % [job number]
kill [pid]
```

- 작업 강제 종료 옵션 `-9`
- ex) `kill -9 57`

하드링크 / 소프트링크

cp

: A파일을 B파일로 복사, 물리적으로 같은 크기를 가진 파일이 두 개가 됨

```
cp [option] [A] [B]
```

- 모든 파일 통째로 복사 `cp -rf * [폴더명]`

ln

: **하드링크** 두 가지 이름이 하나의 물리적 파일을 가리킴

: **소프트링크** 해당 파일로 이동하여 실행하도록 함, 바로가기와 같은 역할, `-s` 옵션

```
ln [option] [A] [B]
```

- 하드링크, 소프트링크 둘 다 A와 B 중 하나의 파일을 수정하면 다른 하나의 파일 이름으로도 접근 가능

우분투 패키지 관리

우분투

: 리눅스 배포판 중 하나

- `apt` 명령어 주로 사용

우분투 패키지 인덱스 정보 업데이트

```
sudo apt-get update
```

패키지 설치

```
sudo apt-get install [package name]
```

패키지 삭제

: **default** 설정파일 제외 삭제

: `--purge` **옵션** 설정파일 포함 삭제

```
sudo apt-get remove [option] [package name]
```

VIM 사용법

VIM

: **vi**에 자동화, 시각화 메뉴 등 추가한 에디터

설치

```
sudo apt-get install vim
```

네 가지 모드

- 일반(명령) 모드: 처음 에디터 시작 시
- 명령(명령행) 모드: 일반모드에서 `:` 입력 시
- 입력(편집) 모드: 일반모드에서 `i` 입력 시
- visual 모드: 일반모드에서 `v` 입력 시

주요 명령어

- 입력
  - `i` 포커스 잡힌 위치에서 편집모드 시작
  - `a` 포커스 잡힌 위치의 다음 칸에서 시작
  - `I` 포커스 잡힌 줄의 맨 앞 위치에서 시작
  - `A` 포커스 잡힌 줄의 맨 뒤 위치에서 시작

- 삭제
  - `x` 커서의 한 문자 삭제, 일반모드에서 실행
  - `dd` 커서가 위치한 한 줄 지우기
  - `2dd` 커서가 위치한 줄과 그 다음 줄, 두 줄 지우기
- 복사/붙여넣기
  - `yy` 커서 위치 한 줄 복사
  - `p` 붙여넣기
- 파일 저장 및 나가기
  - `:w` 현재 오픈된 파일 저장
  - `:q` 현재 오픈된 파일 닫기
  - `:wq` 현재 오픈된 파일 저장 후 종료
  - `:q!` 현재 오픈된 파일 강제 종료
- 검색하기
  - vi 명령모드([ESC] 누른 상태)에서 `/{검색어}` 로 검색 가능
  - `n` 다음 검색어 위치로 이동
  - `N` 이전 검색어 위치로 이동

`vi에디터 명령어` https://blog.lael.be/post/7321

`VIM 꾸미기` https://medium.com/sunhyoups-story/vim-에디터-이쁘게-사용하기-5b6b8d546017

쉘 프로그래밍 예제

1) 간단한 계산기 만들기

요구사항)

1. 두 숫자를 각각 다른 줄로 입력받는다.
2. 사용자는 1, 2, 3, 4의 입력으로 사칙연산 중 하나를 선택할 수 있으며, 사칙연산의 구현은 case 문을 사용한다.

3. 결과창에는 계산 식과 답이 출력된다.
4. 만약 사칙연산에서 잘못된 값을 입력한 경우 "Enter Retry...Next Times"를 출력한다.

- ex) **회색**은 사용자가 직접 입력하는 값

> Enter A: **100**
> Enter B: **20**
> ==========
> (1) + (2) - (3) * (4) /
> ==========
> Enter tour Choice?: **2**
> 100 - 20 = 80

▼ HINT1)

command line 입출력은 \c와 read를 사용한다

```
# 입력값을 A에 저장
echo "Enter A : \c"
read A
```

▼ HINT2)

case 문은 다음과 같이 사용한다

```
case 문자열 in
  경우1) 명령어1;;
  경우2) 명령어2;;
  ...
  *) 명령어
esac
```

▼ ANSWER)

```
#!/bin/sh

echo "Enter A : \c"
read A

echo "Enter B : \c"
```

```
read B

echo "============================================="
echo " (1) + (2) - (3) * (4) / "
echo "============================================="

echo "Enter Your Choice? : \c"
read C

case $C in
     1) echo "$A + $B = `expr $A + $B`" ;;
     2) echo "$A - $B = `expr $A - $B`" ;;
     3) echo "$A * $B = `expr $A \* $B`" ;;
     4) echo "$A / $B = `expr $A / $B`" ;;
     *) echo "Enter Retry...Next Times"
esac
```

2) 나만의 ls 만들기

요구사항)

1. 현재 디렉토리 내에 있는 파일/디렉토리 값을 가져온다(ls).
2. 디렉토리면 “directory ${DirectoryName}”으로 출력한다.
3. 파일이면 “file ${FileName}” 뒤에 파일 내부의 값 3 줄을 출력한다.

- 출력 ex)

> directory Desktop
> directory Downloads
> file files.txt
> asdaf
> asdf
> asdfsdf

▼ HINT1)

for 문 사용법

```
for 변수 in [범위](리스트, 배열 등)
do
```

```
  반복할 작업
done
```

if 문 사용법

```
if [조건문];then
  내용
elif [조건문];then
  내용
...
else
  내용
fi
```

▼ HINT2)

자료형에 관계없이 변수에 저장할 수 있으며, 배열일 경우 인덱싱도 가능하다.

```
names=($(ls ./))
```

```
for i in ${names[*]}
```

▼ HINT3)

test 명령어를 이용하여 디렉토리인지, 파일인지 구분할 수 있다.

- `[ -d ${name} ]`: 디렉토리인지
- `[ -f ${name} ]`: 파일인지

▼ ANSWER)

```
#!/lib/sh

names={$(ls ./)}
for i in ${names[*]}
do
  if [ -d $i ];then
    echo "directory $i"
  elif [ -f $i ];then
    echo "file $i"
    head -3 ./$i
  fi
done
```

맥 사용자들을 위한..

터미널 창 꾸미기

(링크)

그 외…

permission denied / Operation not permitted

1. root 계정의 비밀번호를 부여하고

```
sudo passwd root
# 비번 입력
```

2. root 로 접속해서 마저 진행

```
su -
# 비번 입력
```

변수 이름 커스텀

: alias 사용하기

```
alias ll='ls -al'
alias gc='git commit'
```

ssh 서버 접속

1. **MacOS(링크)**

2. **Windows(링크)**
    - **Putty** 이용(링크)

sh 파일 실행시키기

1. root로 실행(위 참고)
2. 직접 실행

```
/bin/sh {실행파일.sh}
```

3. 파일 사용 권한 변경

```
sudo chmod 777 {실행파일.sh}
```

4. 파일 소유자 변경

```
sudo chown {user}:{group} {실행파일.sh}
```

- 사용자 이름은 `whoami` 로, 그룹은 `groups {user}` 로 알 수 있음