4장

Linux의 핵심적인 기본 명령어

전체 내용

리눅스 파일의 종류와 특징

Directory 다루기

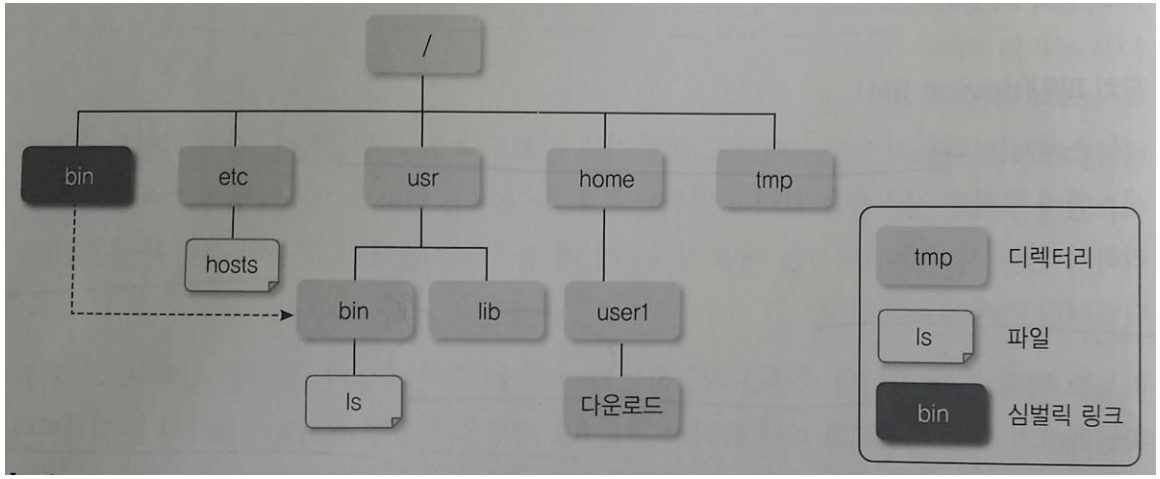
File 다루기

1 – 리눅스 파일의 종류와 특징

- 리눅스 파일의 종류
- 디렉터리 계층 구조이해 하기
- 파일의 절대 경로와 상대 경로 이해하기
- 파일과 디렉터리 이름의 규칙

1 – 리눅스 파일의 종류와 특징

- 리눅스 파일의 종류
  - Linux는 컴퓨터 System과 관련된 정보와 Printer, CDROM과 같은 Device를 모두 File 형태로 관리한다
  - Linux는 이러한 파일들을 효과적으로 관리하기 위해 Directory를 사용한다
  - Linux에서 Directory는 계층적인 구조를 가지고 있다



  - **yum install tree –y**
    tree /home
    tree / -L 1
    tree /etc –L 2

1 – 리눅스 파일의 종류와 특징

- 리눅스 파일의 종류
  - 일반 파일
    - 늘 다루는 일반적인 파일을 말한다
    - 텍스트 파일, 실행 파일, 이미지 파일…
  - Directory
    - 리눅스에서는 Directory도 File로 취급한다
    - Directory file에는 파일이나 또 다른 Directory가 포함되어 있다
  - Symbolic Link
    - 원본 파일을 대신하여 다른 이름으로 파일명을 지정한 것으로서 윈도우의 “바로 가기 " 랑 비슷하다
  - Device File
    - 리눅스에서는 **하드 디스크**, **키보드**와 같은 장치들도 파일로 취급한다
    - 장치 파일은 리눅스 시스템에 장착된 장치들을 관리하기 위해 필요한 특수한 파일을 말한다
    - Device file은 **/dev** 디렉터리 아래에 있다

1 – 리눅스 파일의 종류와 특징

- 리눅스 파일의 종류
  - 리눅스는 윈도우와 달리 파일 확장자가 특별한 의미를 갖는 것이 아니다. 단지 편의성만 제공해줄 뿐이다. 특정한 파일이 어떤 종류인지 확인하려면 **[file 파일이름]**으로 확인할 수 있다
  - 파일의 종류를 알아 보는 명령어: **file**
    - **ls -al**
    - **file** *.bashrc*
    - **mkdir** *webtime*
    - **file** *webtime*
      - **ls –aF**
    - **file** */bin/ls*

```
root@linux200:/home/adminuser# file .bashrc
.bashrc: ASCII text
root@linux200:/home/adminuser# file webtime
webtime: directory
root@linux200:/home/adminuser# file /bin/ls
/bin/ls: ELF 64-bit LSB executable, x86-64,
 (uses shared libs), for GNU/Linux 2.6.32, B
acb1313b08aa23c26, stripped
root@linux200:/home/adminuser#
```

1 – 리눅스 파일의 종류와 특징

- 디렉터리 계층 구조이해 하기
  - **/**
    - 최 상단 디렉터리는 root(/)이다. 모든 파일의 출발점은 root directory이다
    - **ls –F /**
  - **.**
    - 현재 위치하고 있는 디렉터리를 말한다.
    - 상대적인 경로를 지정할 때 주로 사용한다
    - cd ./다운로드
  - **..**
    - 현재 위치하고 있는 곳을 기준으로 바로 위의 상위(부모) 디렉터리를 말한다
    - 상대적인 경로를 지정할 때 주로 사용한다
    - cd ../jesuswithme

1 – 리눅스 파일의 종류와 특징

- 디렉터리 계층 구조이해 하기
  - **Working Directory (. 로 표현)**
    - 리눅스 시스템을 관리할 때 여러 경로로 이동하게 된다. 내가 현재 어느 위치에서 작업을 하고 있는지를 working directory라고 한다
    - 현재 작업하고 있는 디렉터리를 확인할 때는 pwd(present working directory)를 사용한다
      - **pwd**
  - **Home Directory (~ 로 표현)**
    - 로그온 한 사용자가 마음 놓고 이용할 수 있는 디렉터리이다
    - 일반적으로 사용자 계정을 새롭게 생성할 때 그 사용자만이 이용할 수 있는 공간을 배정하는데, 이것을 홈 디렉터리라고 한다
      - **useradd –m –d /home/jesuswithme jesuswithme**
        passwd jesuswithme
        ls /home
    - 어느 위치에서든 자신의 홈디렉터리로 이동하려면
      - **cd ~**

1 – 리눅스 파일의 종류와 특징

- 파일의 절대 경로와 상대 경로 이해하기
  - 파일이 어느 위치에 있는지를 표현하는 것이 Path(경로)이다
  - 경로에는 [절대 경로]와 [상대 경로]가 있다
  - 절대 경로는 항상 **/**(root)로 시작하는 것을 말한다
    - **ls /home/jesuswithme**
    - **cd /tmp**
  - 상대 경로는 현재 위치한 곳에서부터 시작하는 것을 말한다
    - 현재 위치에서 상위로 이동할 때는 **../**를 사용한다
    - **cd ~**
      ls ../home
      cd ../home/jesuswithme
      cd ./다운로드

1 – 리눅스 파일의 종류와 특징

- 파일과 디렉터리 이름의 규칙
  - 파일과 디렉터리 이름을 생성할 때 /은 사용할 수 없다
    - /은 경로명에서 구분자로 사용하기 때문이다
  - 파일과 디렉터리 이름에는 다음과 같은 것만 사용할 수 **있다**
    - 알파벳, 숫자, 붙임표(-), 밑줄(_), 마침표(.)만 사용한다
  - 파일과 디렉터리 이름에는 다음과 같은 것만 사용할 수 **없다**
    - 공백 문자, *, |, “, ‘, @, #, $, %, ^, &
  - 파일과 디렉터리의 영문자 대문자와 소문자를 다른 글자로 취급한다
  - 파일과 디렉터리 이름을 마침표(.)로 시작하면 숨김 파일을 말한다

2 – Directory 다루기

- 현재 위치한 경로 확인하기
- 특정한 Directory로 이동하기
- Directory에 있는 내용 보기
- Directory 만들기
- Directory 삭제하기

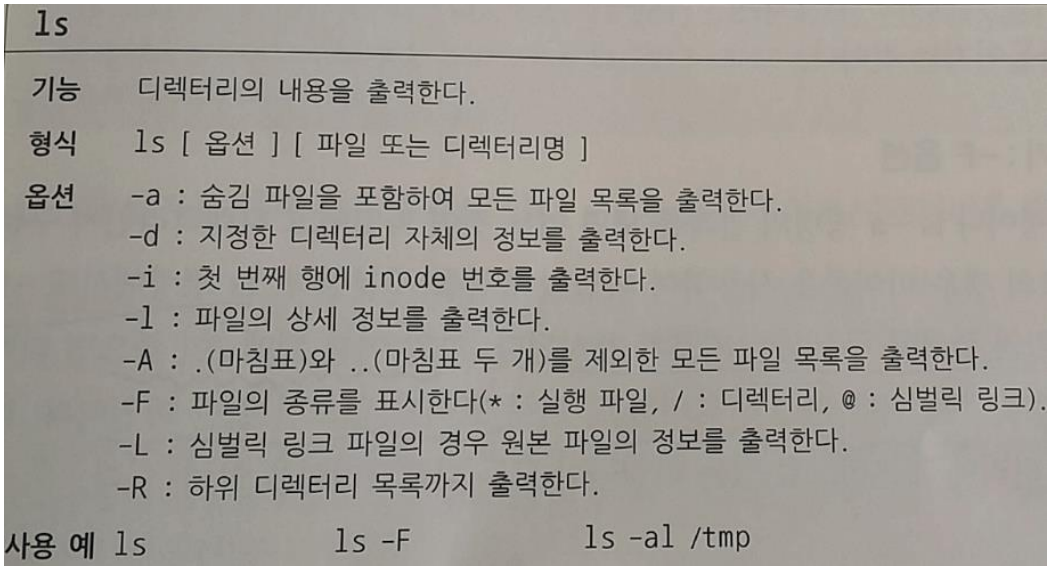
2 – Directory 다루기

- 현재 위치한 경로 확인하기: **pwd**(present working directory)
  - 여러 시스템에 접속하여 동시에 작업을 할 때는 반드시 현재 위치를 확인한 후 작업을 할 필요가 있다.
  - **pwd**

- 특정한 Directory로 이동하기: **cd** (change directory)
  - 자신이 원하는 위치 및 경로로 이동할 때 사용한다
  - **cd /tmp**
    pwd
    cd ~ (##자신의 홈 디렉터리로 이동할 때는 **cd**만 사용해도 된다)
    cd ~/down (##home directory아래에 있는 down 디렉터리로 이동)
    cd ../home/jesuswithme

2 – Directory 다루기

- Directory에 있는 내용 보기: **ls** (list)
  - 리눅스 시스템을 다룰 때 가장 많이 사용하는 명령어이므로 각 옵션에 대하여 정확히 알고 있어야 한다
  - man ls로 정확하게 구문을 확인한다
  - ls와 같은 것이 dir이다.
    - **dir**은 ls의 symbolic link이다
    - **vdir**은 ls –l의 symbolic link이다



ls

기능 디렉터리의 내용을 출력한다.

형식 ls [옵션] [파일 또는 디렉터리명]

옵션
- -a : 숨김 파일을 포함하여 모든 파일 목록을 출력한다.
- -d : 지정한 디렉터리 자체의 정보를 출력한다.
- -i : 첫 번째 행에 inode 번호를 출력한다.
- -l : 파일의 상세 정보를 출력한다.
- -A : .(마침표)와 ..(마침표 두 개)를 제외한 모든 파일 목록을 출력한다.
- -F : 파일의 종류를 표시한다(* : 실행 파일, / : 디렉터리, @ : 심벌릭 링크).
- -L : 심벌릭 링크 파일의 경우 원본 파일의 정보를 출력한다.
- -R : 하위 디렉터리 목록까지 출력한다.

사용 예 ls　　ls -F　　ls -al /tmp

2 – Directory 다루기

- Directory에 있는 내용 보기: **ls** (list)
  - 현재 디렉터리에 있는 내용 확인하기
    - **ls**
  - 숨김 파일까지 모든 것 확인하기: **-a** (-a는 all)
    - **ls –a**
    - 숨김 파일은 주로 사용자 정의 환경 설정이나 응용 프로그램의 환경 설정 파일이다
    - 마침표(.)와 상위 디렉터리(..)도 표시해준다. 이 두 개만 빼고 숨김 파일까지 모두 보고자 하면 **ls –A**를 사용하면 된다
  - 파일 종류 표시하기: **-F**
    - **ls –F**
    - **ls –aF** (##숨김 파일에 대한 종류도 확인할 수 있다)
    - 파일 이름 뒤에 /,@,*를 붙여서 파일의 종류를 말해 준다
    - **/는 directory, @는 Symbolic Link, *는 실행 파일**이고 아무 표시가 없으면 일반적인 파일이다
    - **ls –alF** (=**ll** ##Ubuntu만 가능, CentOS는 ls -l이 ll이다)

2 – Directory 다루기

- Directory에 있는 내용 보기: **ls** (list)
  - 디렉터리 내용 출력하기: **ls 경로**
    - 특정한 경로로 이동하지 않고 디렉터리 내용을 확인할 수 있다
    - **ls /tmp**
    - **ls –F /tmp**
  - 디렉터리의 상세한 정보 출력: **-l** (소문자 L이다. L은 long)
    - 윈도우에서 “자세히 보기 " 와 같다
    - **ls –l workshop.txt**
      -rw-r--r--. 1 root root 0 2015-06-03 10:05 workshop.txt
    - **ls –al**

필드 번호	필드 값	의미
1	d	파일 종류
2	rwxr-xr-x	파일 접근 권한. 파일의 소유자, 그룹, 기타 사용자가 파일을 읽고 수정하고 실행할 수 있는 권한이 어떻게 부여되어 있는지를 보여준다.
3	2	하드 링크의 개수
4	user1	파일 소유자
5	user1	파일이 속한 그룹
6	4096	파일 크기(바이트 단위)
7	2월 5 22:14	파일이 마지막으로 수정된 시간
8	공개	파일 이름

2 – Directory 다루기

- Directory에 있는 내용 보기: **ls** (list)
  - Directory 자체 정보 자세히 보기: **-ld**
    - **ls –l**
    - **ls –ld** (##pwd의 정보)
    - **ls –ld /**
      dr-xr-xr-x. 25 root root 4096 2015-06-03 08:18 /
    - **ls –ld /tmp**

문자	파일 종류
–	일반(정규) 파일
d	디렉터리 파일
l	심벌릭 링크 파일
b	블록 단위로 읽고 쓰는 블록 장치 파일
c	섹터 단위로 읽고 쓰는 문자 장치 파일
p	파이프 파일. 프로세스 간 통신에 사용되는 특수 파일
s	소켓. 네트워크 통신에 사용되는 특수 파일

2 – Directory 다루기

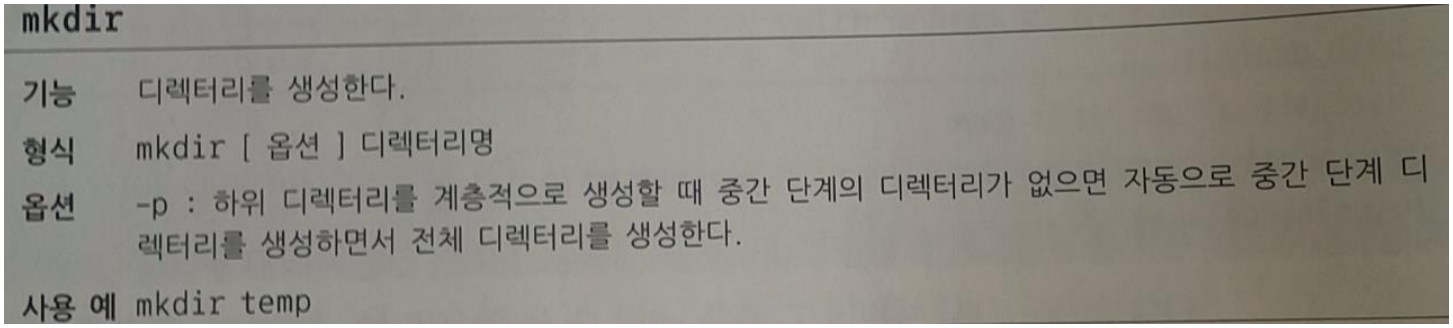
- Directory에 있는 내용 보기: **ls** (list)
  - 특정한 Directory에 내가 찾는 파일이 존재하는지 확인하기
    - **ls 경로**
    - **ls ~/workshop.txt**
    - **ls /home/jesuswithme/project.ppt**
  - 특정한 Directory에 포함된 파일까지 한꺼번에 확인하기: **-R** (=recursive)
    - **ls –R**
    - **ls –alR**
      (## 하위 디렉터리들이 가지고 있는 파일들을 하위 디렉터리별로 자세히 확인)
    - **ls –R /home**
      ls –alR /home
  - 정렬된 파일을 역순으로 보기: **-r** (=reverse)
    - **ls –l**
      ls –lr

2 – Directory 다루기

- Directory에 있는 내용 보기: **ls** (list)
  - 특정한 것을 기준으로 정렬(sort)하여 보기: **--sort=정렬항목**
    - **정렬 항목에는 none, size, time, version, extension**
    - **ls –l**
    - **ls –l --sort=size**
      ls –lr --sort=size (##-r = reverse)
    - **ls –l --sort=time**
      ls –lr --sort=time

2 – Directory 다루기

- Directory 만들기: **mkdir** (##make directory)
  - 접근 권한이 있는 어느 위치에도 디렉터리를 생성할 수 있다
  - 보통 로그온 한 사용자는 자신의 홈 디렉터리에서 디렉터리를 생성한다
  - 접근할 수 없는 디렉터리의 소유자가 특정한 사용자에게 권한을 부여해주면 그 디렉터리에도 디렉터리를 만들 수 있다.



mkdir

기능	디렉터리를 생성한다.
형식	mkdir [옵션] 디렉터리명
옵션	-p : 하위 디렉터리를 계층적으로 생성할 때 중간 단계의 디렉터리가 없으면 자동으로 중간 단계 디렉터리를 생성하면서 전체 디렉터리를 생성한다.
사용 예	mkdir temp

2 – Directory 다루기

- Directory 만들기: **mkdir** (##make directory)
  - 디렉터리 한 개 생성하기
    - **mkdir ~/temp**
  - 동시에 여러 개의 디렉터리 생성하기
    - mkdir student1 student2 student3
  - 디렉터리를 만들 때 경로가 없는 경우에는 경로를 만들면서 생성: **-p**
    - **mkdir /tmp/students/student1** (실패)
    - **mkdir -p /tmp/students/student1** (성공)
      ls –R /tmp
      ls –R ~/temp

2 – Directory 다루기

- Directory 삭제하기: **rmdir** (##remove directory)
  - rmdir로 디렉터리를 삭제하려면 디렉터리 안에 아무 파일도 없어야 한다
  - 내용이 있는 디렉터리 삭제하기: **rm –rf**
  - 디렉터리 삭제하기
    - Empty directory를 한 개만 삭제하기
      rmdir student1
      ls –l
      rmdir /tmp/students/student1
    - Empty directory를 여러 개 동시 삭제하기
      rmdir student2 student3
    - 내용이 들어 있는 Directory를 삭제하기: rm -rf
      mkdir student4
      cp /etc/hosts student4/
      ls -l student4/
      rmdir student4/
      rmdir: failed to remove `student4/': 디렉터리가 비어있지 않음
      rm -rf student4/

3 – File 다루기

- 파일 내용 연속 출력하기: cat (##concatenate)
- 화면 단위로 파일 내용 출력하기: more
- 개선된 화면 단위로 파일 내용 출력하기: less
- 파일 뒷부분 출력하기: tail
- 파일 복사하기: cp
- 파일 이동 및 파일 이름 변경하기: mv
- 파일 삭제하기: rm

3 – File 다루기

- 파일 Link: ln
- 내용이 없는 여러 개의 파일을 한꺼번에 생성하기: touch
- 텍스트 파일 내용 검색하기: grep
- 파일 찾기: find, which, locate, whereis
- Manual과 관련된 명령어: man, whatis, info, apropos
- 표준 입력 값 처리하기: xargs

3 – File 다루기

- 파일 내용 연속 출력하기: **cat** (##concatenate)
  - 짧은 텍스트 파일의 내용을 본다
    - /etc/hosts 파일의 내용을 보기만 하고 편집을 하지 않는다
      cat /etc/hosts
    - 파일의 내용의 한 줄 한 줄에 번호를 붙여서 출력한다(몇 개의 행인지 확인)
      cat –n /etc/hosts
  - **파일들을 연결하여** 화면에 출력하거나 파일로 저장한다
    - /etc/passwd와 /etc/group 파일을 연결하여 화면에 출력
      cat /etc/passwd /etc/group
      cat –n /etc/passwd /etc/group
    - /etc/passwd와 /etc/group 파일을 연결하여 **새로운 파일**로 저장
      cat /etc/passwd /etc/group > users-and-groups.txt
      more users-and-groups.txt

** cat 결과를 역순으로 출력하려면 **tac**
cat /etc/passwd
tac /etc/passwd

3 – File 다루기

- 파일 내용 연속 출력하기: **cat** (##concatenate)
  - 텍스트 파일에 내용을 추가하여 생성하기 (입력할 내용을 파일과 연결)
    - **cat > gospel.txt**
      The beginning of the gospel about Jesus Christ, the Son of God. **ENTER**
      CTRL + D (##저장하기)
  - **기존** 텍스트 파일에 내용을 **첨가**하기 (입력할 내용을 기존 파일과 연결)
    - **cat >> gospel.txt**
      It is written in Isaiah the Prophet. **ENTER**
      CTRL + D (##저장하기)
    - 새로운 파일에 입력하고 기존 파일에 첨가한 내용 확인하기
      cat -n gospel.txt

3 – File 다루기

- 파일 내용 연속 출력하기: **cat** (##concatenate)
  - 기존 파일의 내용을 **다른 파일(새로운 파일 또는 기존 파일)**에 Overwrite 하기
    - **cat gospel.txt > goodnews.txt** (##새로운 파일)
      cat goodnews.txt
      ls –l gospel.txt goodnews.txt (## 파일 크기가 동일(同一))
    - **cat /etc/passwd > goodnews.txt** (##기존 파일)
      cat goodnews.txt (## 파일의 내용이 바뀌었다)
      ls –l gospel.txt goodnews.txt (## 파일 크기가 상이(相異))
  - /etc/hosts 파일의 내용을 **이미 존재하는 다른 파일**(users-and-groups.txt)에 **첨부**하기 (File1을 기존 파일 File2와 연결하여 내용 첨부)
    - **cat /etc/hosts >> users-and-groups.txt**
      tail users-and-groups.txt

3 – File 다루기

- 화면 단위로 파일 내용 출력하기: **more**
  - cat 명령어로 용량이 큰 텍스트 파일 내용을 보면 자동 스크롤이 되어 앞의 내용을 볼 수 없다
  - 텍스트 파일의 내용이 많을 경우 **한 페이지씩** 화면에 출력하여 보여주는 것이 more이다
    - **more** /etc/services
    - cat /etc/service **| more**

  - More에 사용되는 옵션
    - **10개 행씩 화면에** 출력
      cat –n /etc/passwd
      more -10 /etc/passwd (또는 cat /etc/passwd | more -10)
      (## 여기서 SPACE를 누르면 10개씩 화면에 출력)

    - 파일의 **20번째 행부터** 그 이후로 출력
      cat –n /etc/passwd
      more +20 /etc/passwd (또는 cat /etc/passwd | more +20)

3 – File 다루기

- 화면 단위로 파일 내용 출력하기: **more**
  - **more** /etc/services 를 실행한 후 more 명령어의 내부 명령어
    - **[space]** : 다음 페이지로 이동한다
    - **q** 또는 **Ctrl + z** : more 명령어 작업을 종료한다
    - **v** : vi, vim 편집기 프로그램을 호출한다
    - **!** : sub shell로 빠져 나가서 원하는 명령어를 실행할 수 있다. 명령어 결과가 나타난 후에는 계속 more 작업이 진행된다
    - **=** : 현재 행 번호를 보여준다

3 – File 다루기

- 개선된 화면 단위로 파일 내용 출력하기: **less**
  - less는 more는 동일하지만 less가 조금 더 개선된 기능을 가지고 있다
  - Linux: less, more 모두 사용 가능, 하지만 Unix: more만 지원
  - tty인 terminal에서 작업할 때 less와 more 모두 다음 기능이 가능하다
    - 이전 화면으로 이동: **ctrl + b**
    - 다음 화면으로 이동: **ctrl + f**
  - 하지만 다음 기능은 less만 가능하고 more는 지원되지 않는다
    - 한 줄씩 다음 행으로 이동: **j**
    - 한 줄씩 이전 행으로 이동: **k**
    - 화살표 사용
  - Ctrl + F2를 눌러서 Terminal로 빠져 나간다
    - more /etc/services
    - less /etc/services
      ctrl + f
      ctrl + b
      j

3 – File 다루기

- 파일 뒷부분 출력하기: **tail**
  - 파일의 뒷부분의 10개 줄만 기본적으로 출력한다
    - **tail /etc/services**
  - 기존 파일에 새롭게 추가된 것만 보고자 할 때 사용하면 편리하다
  - 10개 행 대신 지정한 숫자만큼 뒷부분부터 출력하기
    - **tail -n 15 /etc/services**
  - **계속 업데이트되는 로그 파일**을 뒷부분부터 계속 보기
    - -f 옵션은 tail 명령이 무한 반복되는 것을 말한다. 이것을 중지하기 위해서는 **ctrl +c**
    - **tail –f error.log**
  - /var/log에 로그 파일들이 저장되어 있다
    - **tail /var/log/yum.log** (##최근에 설치된 프로그램 확인하기)
    - **tail /var/log/apt/history.log** (##Ubuntu에서 최근에 apt-get 명령어 확인)
  - tail의 반대 명령어는 head이다
    - **head /etc/services**
    - **head –n 15 /etc/services**

3 – File 다루기

- 파일 복사하기: **cp**
  - 파일과 디렉터리를 복사할 때 사용한다
    - cp 옵션 file1/directory1 file2/directory2
    - 옵션이 –i(=interactive)인 경우에는 대화형으로 진행한다
    - 옵션이 **–r**(=recursive)인 경우에는 **원본 디렉터리의 하위 내용까지 몽땅 복사한다**
  - 두 인자(source, destination)가 모두 File인 경우
    - **cp /etc/hosts text1** (##text1은 새로운 파일)
      ls –l text1
      cat text1
    - **cp /etc/services text1** (##text1은 내용이 있는 기존 파일로서 cp 작업을 하면 기본 파일 내용은 없어지고 새로운 내용으로 채워진다)
      ls –l text1

```
[root@centos ~]# ls -l text1
-rw-r--r--. 1 root root 158 2015-06-04 00:36 text1
[root@centos ~]# cat text1
127.0.0.1   localhost localhost.localdomain localhost4 l
::1         localhost localhost.localdomain localhost6 l
[root@centos ~]# cp /etc/services text1
cp: overwrite `text1'? yes
[root@centos ~]# ls -l text1
-rw-r--r--. 1 root root 641020 2015-06-04 00:38 text1
[root@centos ~]#
```

3 – File 다루기

- 파일 복사하기: **cp**
  - Source는 파일, Destination이 디렉터리인 경우
    - 동일한 파일 이름으로 복사가 된다
    - 만약 다른 파일 이름을 지정하면 다른 이름으로 복사된다
    - **mkdir temp**
      cp text1 temp/
    - **cp text1 temp/text2**
      ls temp/

```
[root@centos ~]# ls -l temp/
합계 1256
-rw-r--r--. 1 root root 641020 2015-06-04 00:45 text1
-rw-r--r--. 1 root root 641020 2015-06-04 00:46 text2
[root@centos ~]#
```

    - 파일을 복사할 때는 대상 폴더에 쓰기 권한이 있어야 한다
      cp text1 /etc

```
[localuser1@centos ~]$ cp text1 /etc
cp: cannot create regular file `/etc/text1': 허가 거부
[localuser1@centos ~]$
```

3 – File 다루기

- 파일 복사하기: **cp**
  - Source는 **여러 개의 파일**, Destination이 **디렉터리**인 경우
    - 즉, 한 번에 여러 개의 파일을 하나의 디렉터리도 복사할 수 있다
    - **cp /etc/hosts /etc/services temp/**
  - Directory를 복사하기: **-r 옵션 사용**
    - 기존 directory의 내용을 새로운 directory로 복사한다
    - File이 아닌 Directory를 복사할 때는 반드시 **–r** 옵션을 사용해야 하며, 하위 내용까지 몽땅 복사된다(recursive)
    - **cp temp/ temp2/ (## 실패)**
    - **cp –r temp/ temp2/** (##성공-디렉터리 통째로 복사)
      ls –l temp2/
    - **cp -r temp/* temp2/** (temp/ 안에 있는 모든 파일과 디렉터리를 모두 복사)
    - **cp temp/* temp2/** (temp/안에 있는(루트 Only) 파일들만 모두 복사)

3 – File 다루기

- 파일 복사하기: **cp**
  - 파일을 interactive하게 복사하기: **-i옵션 사용**
    - Destination file이 이미 존재하는 경우에만 Overwrite할 것인지 묻도록 하기
    - **cp -i /etc/hosts text1**

```
[localuser1@centos temp]$ cp -i /etc/hosts text1
cp: overwrite `text1'? y
```

3 – File 다루기

- 파일 이동 및 파일 이름 변경하기: **mv**
  - **동일한 Directory내에서 이동(이름 변경의 결과를 가져온다)**
    - File1을 File2로 이동하는 것은 Files1의 **이름을 변경**하는 것이다
    - 만약 목적지 파일인 File2가 존재하지 않는다면 원본 파일인 File1이 목적지 위치를 **이동**하게 되는 셈이다. 결과적으로 기존 **파일 이름이 변경**되는 셈이다
      - **mv text1 data1** (##data1은 기존에 없던 새로운 파일)
        ls –l (##2개의 파일 크기를 보면 동일하다)
    - 목적지 파일인 File2가 존재하면 원본 파일인 File1의 내용으로 File2를 Overwrite하게 된다. 결과적으로 File1이 File2로 **이름이 변경**되는 것이다
      - **mv data1 gospel.txt** (##gospel.txt는 기존 파일이고, 두 파일의 크기가 다르다)
        ls –l (##gospel.txt의 내용이 data2이 것으로 변경되었기에 원래 파일 크기랑 달라졌다)

3 – File 다루기

- 파일 이동 및 파일 이름 변경하기: **mv**
  - **다른 Directory로 이동(이름 변경 유무는 입력하는 내용에 따라 다르다)**
    - File1을 다른 디렉터리로 이동할 때 **디렉터리 이름만 지정하면** 원본 파일 이름으로 이동한다
      - **ls –l**
        mv gospel.txt temp/
        ls –l temp/
    - File1을 다른 디렉터리로 이동할 때 **디렉터리 이름과 목적지 파일 이름을** 모두 입력하면 입력한 파일 이름으로 이동하게 된다(원본과 동일 또는 상이하게 입력할 수 있다)
      - **mv goodnews.txt temp/GoodNews.txt**
        ls –l temp/ (##파일 이름이 변경되어 이동되었다)
    - 쓰기 권한이 없는 위치로 이동은 불가하다
      - **mv temp/GoodNews.txt /etc**
    - 원본 위치에 여러 개의 파일을 사용하여 목적지로 한꺼번에 이동하기
      - **cd temp/**
        mv text1 text2 ../ (## 2개의 파일을 상위 디렉터리로 이동)

3 – File 다루기

- 파일 이동 및 파일 이름 변경하기: **mv**
  - **Directory1을 Directory2로 이동**
    - mv의 인자를 모두 Directory로 하면 된다
    - **Directory 이름을 변경하고자 한다면** 목적지 디렉터리를 새로운 이름의 디렉터리를 사용하면 된다
      - **cd ~**
        ls –lF
        mv temp/ temp2/
        ls –lF
    - 원본 디렉터리를 기존 다른 디렉터리(temp3/)로 이동
      - **mkdir temp3/**
        mv temp2/ temp3/
        ls temp2/ (##temp2가 안 보임)
        ls temp3/ (## temp3/에 temp2가 이동되어 있음)

3 – File 다루기

- 파일 삭제하기: **rm**
  - 파일을 삭제할 때는 항상 조심해야 한다. 파일을 삭제하면 휴지통으로 들어 가지 않으므로 복구할 수 없다
  - 파일 삭제하기
    - **rm users-and-groups.txt** (##진짜 삭제할 것인지 물어 본다)
    - **rm –f workshop.txt** (##그냥 삭제한다)
  - 디렉터리 삭제하기: **-r** 옵션 사용
    - **cp –r temp3/ temp4/** (##temp4/를 하나 더 만든다)
      rm –r temp3/ (##디렉터리 안에 있는 파일과 폴더들을 일일이 삭제할 것인지 물어 본다.)
    - **rm –rf temp4/** (##그냥 삭제한다 / **파일 및 폴더에 모두 사용 가능;강추)**
  - 디렉터리에 파일이나 또 다른 디렉터리가 없는 경우에는 rmdir을 사용해도 된다
  - 대화형으로 삭제하려고 할 때: **-i** 옵션 사용
    - **rmdir -i directory1** (##비어 있는 디렉터리)
    - **rm -i file1**
    - **rm -ri dirrectory3** (##비어 있지 않는 디렉터리)

3 – File 다루기

- 파일 Link: **ln**
  - 파일 link는 기존에 있는 파일에 새로운 파일명을 붙이는 것이다
  - 복잡한 경로에 접근할 때 짧게 줄인 link를 사용하면 좋다
  - Link에는 Hard Link와 Symbolic Link가 있다
    - Hard Link: 기존 파일에 새로운 파일을 추가로 생성하는 것
    - Symbolic Link: 원본 파일을 가리키는 새로운 파일을 생성하는 것
  - 리눅스에서 파일은 **[파일명 + inode + 데이터 블록]**으로 되어 있다
    - 파일명: 사용자가 파일에 접근할 때 사용하는 것
    - inode: 외부적으로는 **색인 번호**로 표시하고, 내부적으로는 **상세한 파일 정보**(파일의 종류, 크기, 소유자, 파일 변경시간, 파일명)와 **데이터 블록의 주소**가 저장되어 있다
    - 데이터 블록: 실재로 데이터가 저장된 장소
    - **ls -i**를 했을 때 파일 이름 앞에 있는 것이 inode 번호이다. **파일 이름이 다르더라도 inode 번호가 동일하면 같은 파일이다** (옵션 –i는 inode)

```
[root@centos ~]# ls -i
786594 GoodNews.txt        786590 temp3
786441 anaconda-ks.cfg     786596 text1
786434 install.log         786597 text2
```

3 – File 다루기

- 파일 Link: **ln**
  - Hard Link 만들기: **ln file1 file2**
    - 리눅스에서는 특정한 파일에 **여러 개의 파일 이름**을 붙일 수 있다. 이것을 Hard Link라고 한다
      - 저장된 데이터는 하나이지만 그 데이터에 대한 파일 이름은 여러 개
    - 파일이 같은지 다른지 구분하는 방법은 파일들의 inode를 확인하여 같은 색인 번호이면 같은 파일이고, 색인 번호가 다르면 다른 파일이다
    - 새로운 Hard Link(동일한 파일)와 새로운 파일을 생성한다(다른 파일)
      - **ls -l text1**
        -rw-r--r--. **1** root root 641020 2015-06-04 00:45 text1
        ln text1 text1.ln
        ls -l text1*
        -rw-r--r--. **2** root root 641020 2015-06-04 00:45 text1
        -rw-r--r--. **2** root root 641020 2015-06-04 00:45 text1.ln
      - **cp text1 text1.cp**
        ls –l text1*
        -rw-r--r--. **2** root root 641020 2015-06-04 00:45 text1
        -rw-r--r--. **1** root root 641020 2015-06-04 09:59 text1.cp

3 – File 다루기

- 파일 Link: **ln**
  - Hard Link 만들기: ln
    - text1, text1.ln, text1.cp의 파일이 같은 파일인지 다른 파일인지 구분하기
      - **ls -li text1***
        786596 -rw-r--r--. 2 root root 641020 2015-06-04 00:45 text1
        786592 -rw-r--r--. 1 root root 641020 2015-06-04 09:59 text1.cp
        786596 -rw-r--r--. 2 root root 641020 2015-06-04 00:45 text1.ln
    - Text1, text1.ln은 inode가 같은 번호이므로 **같은 파일**이다
    - Text1을 복사한 text1.cp파일은 inode가 text1과 다르므로 **다른 파일**이다
    - Text1, text1.ln에서 2라는 뜻은 2개의 link가 있다는 뜻이다
    - Text1.cp에서 1이라는 것은 link가 하나만 있다는 것.즉, 다른 파일은 없다
    - Text1.cp는 복사한 파일이기 때문에 text1과는 별개로 독립적인 파일이다
    - Nano, gedit, vi, vim 같은 편집 프로그램을 사용하여 text1 파일을 변경하면 text1.ln 파일의 내용도 변경되어 있음을 알 수 있다. 그 이유는 text1과 text1.ln은 동일한 inode 색인 번호를 가지고 있어서 동일한 데이터 블록 주소를 가지기 때문에 같은 파일이기 때문이다
    - 하지만 text1.cp파일을 열어보면 여기는 변경되어 있지 않다. 그래서 복사한 파일과 Link 파일의 차이를 알 수 있다.

3 – File 다루기

- 파일 Link: **ln**
  - Hard Link 만들기: ln
    - 파일을 삭제하면 하드링크 값을 하나씩 줄이는 셈이 된다. 하드링크 값이 0이 되면 그제서야 해당 inode와 데이터 블록을 완전히 삭제하는 것이다

      rm –rf text1

      ls -li text1*

      786592 -rw-r--r--. 1 root root 641020 2015-06-04 09:59 text1.cp

      786596 -rw-r--r--. **1** root root 641020 2015-06-04 00:45 text1.ln

3 – File 다루기

- 파일 Link: **ln**
  - Symbolic Link 만들기: **ln -s**
    - Symbolic Link는 윈도우의 [바로 가기]랑 비슷하지만 동일한 것은 아니다
    - **ln -s text1 text1.sl**
      ls -li text1*
      786598 -rw-r--r--. 1 root root 641020 2015-06-04 10:29 text1
      786599 lrwxrwxrwx. 1 root root 5 2015-06-04 10:30 text1.sl -> text1
    - Symbolic Link의 특징
      - 원본 파일과는 다른 독립적인 파일이다(inode 색인번호가 다르다)
      - 파일의 크기를 보면 용량이 작다. 원본 파일의 위치만을 가지고 있다
      - Symbolic Link 파일을 열어 보면 원본 파일 내용이 보인다
      - 원본 파일이 삭제된 상태에서 심볼릭 파일을 열면 오류가 발생한다
      - ls –l text1.sl을 하면 파일 종류가 소문자 L로 표현되고, 원본 파일도 보여준다(-> text1)
      - 원본 파일 및 원본 디렉터리와 다른 파일 시스템에 위치할 수 있다
      - 복사한 symbolic link 파일을 업데이트하면 원본이 동시에 수정되지 않음
    - Hard Link의 약점
      - 하드 링크는 파일에만 링크를 생성하고 디렉터리에는 생성할 수 없다
      - 하드 링크는 다른 파일 시스템에 파일에 대한 링크를 걸 수 없다

3 – File 다루기

- 내용이 없는 여러 개의 파일을 한꺼번에 생성하기: **touch**
  - 내용이 없는 파일을 생성한다
    - **touch newfile**
      touch newfile1 newfile2
      ls -l newfile*
      -rw-rw-r--. 1 adminuser adminuser 0 2015-10-11 22:24 newfile
      -rw-rw-r--. 1 adminuser adminuser 0 2015-10-11 22:33 newfile1
      -rw-rw-r--. 1 adminuser adminuser 0 2015-10-11 22:33 newfile2
      (##크기가 0이므로 내용 없음)
  - 내용이 없는 파일을 여러 개를 동시에 생성한다
    - **touch {A..Z}**
    - **touch {1..20}**
    - **touch {1..1000}.txt**
    - **touch a{1..50}**
    - **touch gospel{1..1000}**
    - **touch {1..10}{1..25}**
  - 특정한 위치에 쓰기 권한이 있는지 확인할 때 사용한다

3 – File 다루기

- 빈 파일 만들기 및 파일 수정 시간 변경하기: **touch**
  - 기존 파일(existingfile)의 **수정 시간** 변경하기
    - **cat > existingfile**
      How is it going?
      ls -l existingfile
      -rw-rw-r--. 1 adminuser adminuser 17 2015-10-11 22:37 existingfile
    - **touch existingfile**
      ls -l existingfile
      -rw-rw-r--. 1 adminuser adminuser 17 2015-10-11 **22:38** existingfile
  - jsp에서 include 된 jsp 파일을 수정하더라도 서버가 이를 인식 못하고 컴파일이 안되는 경우가 발생될 때
    - 해당 jsp 를 include하고 있는 jsp 파일도 수정해줘야 인식이 된다.
    - 이때 jsp파일 수정을 안하고 아래처럼 명령어를 실행하면 해당 파일 수정 일자가 현재로 모두 바뀌기 때문에 서버가 jsp 파일들을 재컴파일을 할 수 있게 된다.
    - **find ./ -name '*.jsp' | xargs touch**

3 – File 다루기

- 빈 파일 만들기 및 파일 수정 시간 변경하기: **touch**
  - 기존 파일(existingfile)의 Time Stamp 정보 보기: **stat 명령어 사용**
    - **stat existingfile**
      Access: 2015-10-11 **22:38:51**.719208202 +0900
      Modify: 2015-10-11 **22:38:51**.719208202 +0900
      Change: 2015-10-11 **22:38:51**.719208202 +0900
  - existingfile의 **Access 시간 변경하기**
    - **touch -a existingfile**
    - **stat existingfile**
      Access: 2015-10-11 **22:45:32**.967208201 +0900
      Modify: 2015-10-11 **22:38:51**.719208202 +0900
  - existingfile의 **Modification 시간 변경하기**
    - **touch -m existingfile**
    - **stat existingfile**
      Access: 2015-10-11 **22:45:32**.967208201 +0900
      Modify: 2015-10-11 **22:50:46**.392208202 +0900
      (## Change는 Access나 Modify를 변경하면 같이 변경된다. 즉, Access에 변경이 있으면 Change에 Access와 동일한 시간이 설정되어서 Access가 변경된 것이라고 알려주는 역할을 한다)

3 – File 다루기

- 빈 파일 만들기 및 파일 수정 시간 변경하기: **touch**
  - existingfile을 특정한 시간으로 설정하여 변경하기: **-t 201408301407**
    - **touch -t 201408301407 existingfile**
    - **stat existingfile**
      Access: 2014-08-30 14:07:00.000000000 +0900
      Modify: 2014-08-30 14:07:00.000000000 +0900
      Change: 2015-10-11 23:11:36.381208201 +0900
      (## Current time이 아닌 특정한 시간으로 변경하면 Access, Modify가 동시에 변경되고 Change는 원래 값을 유지한다)
  - 다른 파일의 시간을 복사하여 오기: **-r 다른파일**
    - **stat test1**
      Access: 2015-10-11 22:17:10.568208202 +0900
      Modify: 2015-10-11 22:14:09.305208202 +0900
      Change: 2015-10-11 22:17:10.568208202 +0900
    - **touch existingfile -r test1**
    - **stat existingfile**
      Access: 2015-10-11 22:17:10.568208202 +0900
      Modify: 2015-10-11 22:14:09.305208202 +0900
      Change: 2015-10-11 23:18:02.932208202 +0900

3 – File 다루기

- 텍스트 파일 내용 검색하기: **grep**
  - 간단하게 특정한 문자열로 검색할 수도 있고 정규 표현식을 사용하여 세부적으로 검색할 수도 있다
    - **grep root /etc/passwd**
    - **cat /etc/passwd | grep root**
    - **cat /etc/passwd | grep –n root**
    - **grep –n unix ~/*.txt**
    - **cp /etc/services data**
      grep DHCP data
      dhcp-failover 647/tcp # DHCP Failover
      dhcp-failover 647/udp # DHCP Failover
    - **cat data | grep DHCP**
    - **cat data | grep -n DHCP**
      1437:dhcp-failover 647/tcp # DHCP Failover
      1438:dhcp-failover 647/udp # DHCP Failover

3 – File 다루기

- 파일 찾기:
  - **find** [경로] [옵션] [찾을 파일명]
    - 리눅스의 디렉터리 계층 구조에서 자신이 원하는 파일을 찾는다
    - Grep은 파일 내용을 찾고, find는 파일 생성 일자, 이름, 파일 소유자 등 다양한 조건으로 특정한 파일을 찾는다
    - 파일 이름별로 찾기: **-name** 옵션
      - **find /bin -name** ls
        /bin/ls
      - **find /home -name** text1
        /home/localuser1/text1
        /home/localuser1/yslee/temp/text1
    - 특정한 사용자 계정이 소유자인 파일을 찾고자 할 때: **-user** 옵션
      - **find /home -user** localuser1
    - /tmp 아래에 있는 localuser1 소유의 파일을 전부 찾아서 삭제하기:
      -exec rm –rf {} \; 옵션 또는 **-ok rm –rf {} \;** 옵션(묻는 화면 나옴)
      - find /tmp -user localuser1
        find /tmp -user localuser1 -exec rm -rf {} \;
        find /tmp -user localuser1
      - **find /var/log -name "*.log" -exec grep linux {} \; -print**

3 – File 다루기

- echo $PATH에 설정된 디렉터리에 저장된 **명령어만 찾기**: **which**
  - **which grep**
  - **which –a ls**

- 고속으로 데이터베이스에 저장된 파일 찾기: **locate**
  - touch peace.file
    locate peace.file (실패)
  - **updatedb**
    locate peace.file (성공)
  - 파일 찾는 속도가 가장 빠르다

- 명령어 파일만 찾으며, 그 명령어의 source, man 페이지 파일까지 검색: **whereis**
  - **whereis pwd**
    - **which pwd**
    - **whatis pwd**
    - **man pwd**

3 – File 다루기

- Manual과 관련된 명령어: **man, whatis, info, apropos**
  - 명령어에 대한 설명서(manual) 찾아보기: **man**
    - **man ls**
    - **man pwd**
  - Manual의 section 별로 찾기: **whatis**
    - **whatis crontab**
      man 5 crontab
  - Manual보다 더 자세하게 설명서 찾아보기: **info**
    - **info ls**
      man ls (##이 두 개를 비교해볼 것)
  - 특정한 string을 가지고 있는 명령어 찾기: **apropos (=man –k)**
    - **apropos pwd**
      man –k pwd
    - **apropos samba**

3 – File 다루기

- 표준 입력 값 처리하기: **xargs**
  - xargs은 표준 입력 값(stdin)을 받아서 /bin/echo 명령어를 처리한다
  - xargs는 주로 *command1* **| xargs** *command2* 형식으로 사용하며, command1은 주로 **find**이고, command2는 **grep, rm, cp**을 사용한다
  - xargs의 사용 예제
    - 입력 값을 받아서 /bin/echo 실행하기
      - xargs
        Hi,
        Welcome to this class of Linux Basics.
      - ctrl + d

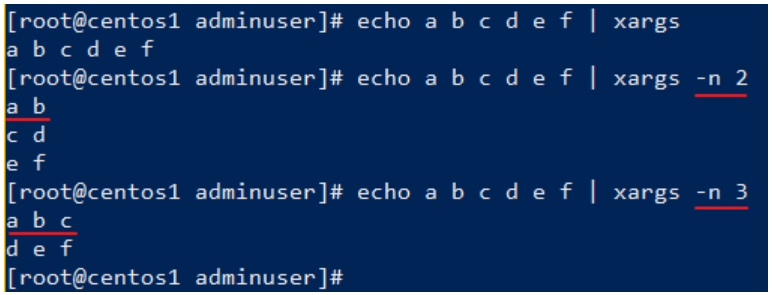
```
[root@centos1 adminuser]# xargs
Hi,
Welcome to this class of Linux Basics
Hi, Welcome to this class of Linux Basics
[root@centos1 adminuser]#
```

    - 입력 값을 받아서 /bin/echo 실행하기-**구분자(delimiter) -d 옵션 사용**
      - xargs **-d\n**
        Hi,
        Welcome to this class of Linux Basics.
      - ctrl + d

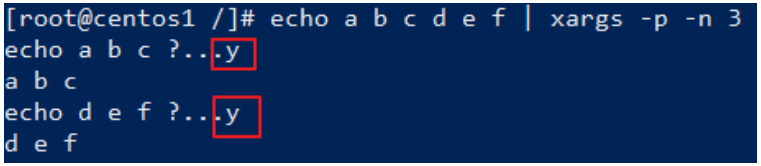
```
[root@centos1 adminuser]# xargs -d\n
Hi,
Welcom to this class of Linux Basics.
Hi,
Welcom to this class of Li ux Basics.
```

3 – File 다루기

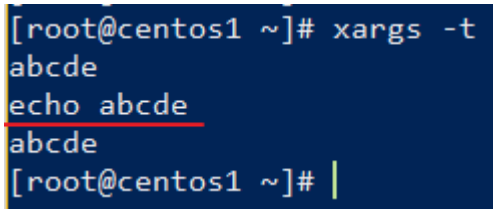
- 표준 입력 값 처리하기: **xargs**
  - xargs의 사용 예제
    - -n 옵션(number)을 사용하여 한 라인 당 결과(수량) 제한하기
      - echo a b c d e f | xargs
      - echo a b c d e f | xargs **-n 2**
      - echo a b c d e f | xargs **-n 3**
      - **echo {0..9} | xargs -n 2**



    - -p 옵션(prompt)을 사용하여 사용자에게 yes, no로 응답 요구하기
      - echo a b c d e f | xargs **-p -n 3**



    - -t 옵션(verbose)을 사용하여 입력한 명령어를 보여 준 후 결과 실행하기
      - xargs **-t**
        abcd
      - ctrl + d



3 – File 다루기

- 표준 입력 값 처리하기: **xargs**
  - xargs의 사용 예제
    - find 명령어와 xargs를 함께 사용하면 강력하게 처리할 수 있다
    - 확장자가 .c로 끝나는 파일만 찾아서 **삭제하기: rm -rf**
      - **touch** abc.c def.c ghi.txt jkl.txt "mn o.txt"
      - find . -name "*.c"
        find . -name "*.c" **| xargs rm -rf**
        find . -name "*.c"
    - 확장자가 .txt로 끝나는 파일만 찾아서 삭제하기
      - find . -name "*.txt" | xargs rm -rf
        find . -name "*.txt" **| xargs**
    - 파일 이름에 공백이 있는 글자도 처리하기: **-print0 | xargs -0**
      - find . -name "*.txt" **-print0 | xargs -0 rm -rf**
        find . -name "*.txt"

```
[adminuser@centos1 ~]$ touch abc.c def.c ghi.txt jkl.txt "mn o.txt"
[adminuser@centos1 ~]$ find . -name "*.c"
./abc.c
./def.c
[adminuser@centos1 ~]$ find . -name "*.c" | xargs rm -rf
[adminuser@centos1 ~]$ find . -name "*.c"
[adminuser@centos1 ~]$ find . -name "*.txt" | xargs rm -rf
[adminuser@centos1 ~]$ find . -name "*.txt" | xargs
./mn o.txt
[adminuser@centos1 ~]$ find . -name "*.txt" -print0 | xargs -0 rm -rf
[adminuser@centos1 ~]$ find . -name "*.txt"
[adminuser@centos1 ~]$
```

3 – File 다루기

- 표준 입력 값 처리하기: **xargs**
  - xargs의 사용 예제
    - 각 파일에 기록된 문자를 찾기 위해서 find 명령어와 **xargs grep 문자열** 사용하기- **xargs grep 사용하기**
    - /etc/services 파일에서 ssh 문자가 들어간 행 찾기
      - find /etc -name services | xargs grep "ssh"
      - find /etc -name services | xargs grep "^ssh"
    - 확장자가 .sh로 끝나는 파일만 찾아서 sh 문자가 들어가 행 찾기
      - find / -name "*.sh" -print0 | xargs -0
      - find / -name "*.sh" -print0 **| xargs -0 grep "sh"**
      - **이 예제는 특정한 파일에 포함된 문자를 찾을 때 유용하다**
    - 특정한 파일 내용이 몇 라인인지 확인하기: **wc -l**
      - ls /etc/services **| xargs wc -l**
      - ls /etc/hosts | xargs wc -l
      - ls -1 *.sh | xargs wc -l

3 – File 다루기

- 표준 입력 값 처리하기: **xargs**
  - xargs의 사용 예제
    - xargs를 사용할 때는 한 번에 하나의 argument를 처리하는 것이 보통이다
    - {}은 default argument list marker로서 **한 번에 둘 이상의 argument**를 처리하는 것이다
    - {} 대신에 file, $ 등등을 사용할 수 있다
    - {}을 사용할 때는 반드시 **-I {}**로 사용한다
    - 확장자가 .sh인 파일을 찾아서 backup.scripts 디렉터리로 이동/복사하기
      - find . -name "*.sh" -print0 | xargs -0 -I {} **mv {} ~/backup.scripts**
      - $ find . -name "*.sh" -print0 | xargs -0 -I file **cp file ~/backup.scripts**