쉘, 프로세스, 사용자, 유틸리티 명령어

Hadoop

Byeongjoon Noh

powernoh@sch.ac.kr

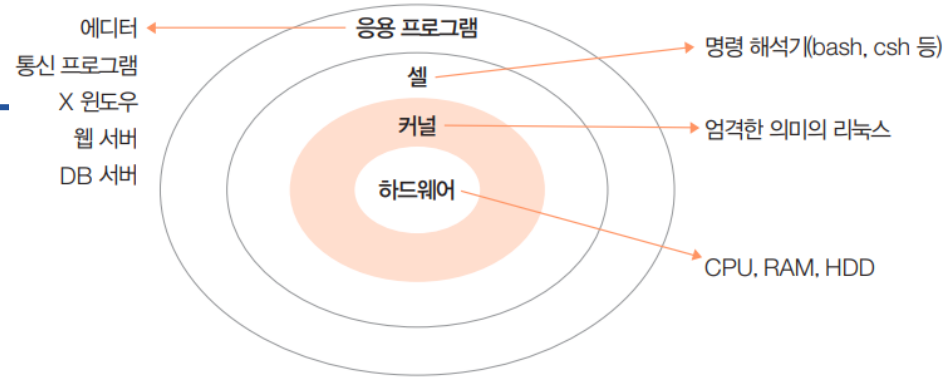
SCH SOON CHUN HYANG UNIVERSITY

Most of the slides are available on Senseable AI Lab homepage: https://sailab.space/courses/

Contents

1. 쉘의 활용

Shell의 개념



- Linux Shell
  - 명령어와 프로그램을 실행할 때 사용하는 인터페이스
    - 사용자와 커널 사이의 중간 역할
  - 사용자가 입력한 명령을 해석하여 커널에 전달하거나, 커널의 처리 결과를 사용자에게 전달하는 역할
  - Ubuntu에서는 기본적으로 bash (Bourne Again Shell)을 이용함

- Shell의 종류
  - 본쉘, 콘쉘, C쉘, 배시쉘, 로그인쉘, 서브쉘 …

Shell의 개념

- Bash shell의 특징
  - alias (명령 단축) 기능
  - history 기능 (↑ 또는 ↓)
  - 연산 기능
  - Job control 기능
  - 자동 이름 완성 기능 (Tab)
  - 프롬프트 제어 기능
  - 명령 편집 기능

31. 다음 설명에 해당하는 셸의 기능으로 알맞은 것은?

> 기존에 실행한 명령들을 위/아래 방향키를 사용하여 검색 및 편집하여 특정 명령을 반복해서 수행할 수 있다.

① 명령행 완성 기능　② 명령행 편집 기능
❸ 명령어 히스토리 기능　④ 명령어 alias 기능

32. 다음 중 현재 사용 가능한 셸 목록 정보가 저장된 파일명으로 알맞은 것은?

① /etc/passwd　❷ /etc/shells
③ /etc/login.defs　④ /etc/default/useradd

33. 다음 설명에 해당하는 셸로 알맞은 것은?

> 1989년 브라이언 폭스가 GNU 프로젝트를 위해 개발한 셸로 명령 히스토리, 명령행 편집 등 다양한 기능을 지원한다.

① ksh　② tcsh
❸ bash　④ dash

쉘의 명령문 처리 방법

- 쉘 명령문의 형식
  - 명령 [옵션] [인자]

```
# ls -l
# rm -rf /mydir
# find . / -name "*.conf"
```

- 쉘 특수 문자의 종류

메타문자	기능	예제
;	한 줄에 여러 개의 명령 입력	$ date;cal;ls
*	임의의 문자 또는 문자들	$ ls h*
?	임의의 한 문자	$ ls dir?
[ ]	한 문자 위치를 위한 문자의 범위 표시	$ ls [a-f]*
>, >>, <	입출력 방향 전환	$ ls > ls.out
|	명령어 파이핑	$ ls –l /etc | more
~	홈 디렉토리	$ cd ~user1
-	이전 작업 디렉토리	$ cd -
‘ ‘	모든 쉘 문자 무시	$ echo ‘$SHELL’
“ “	$, `, \를 제외한 모든 쉘 문자 무시	$ echo “$SHELL”
` `	쉘 명령 수행	$ echo `date`
\	특수문자 기능 제거	$ echo “\$SHELL”

쉘 특수 문자 사용 예제

- ; 명령 연결
  - `$ pwd ; touch a.txt b.txt; ls`
- * 임의의 문자
  - `$ ls *.txt`
- ? 임의의 한 문자
  - `$ touch ab.txt`
  - `$ ls ?.txt`
- [] 범위 내의 한 문자
  - `$ ls [ab].txt`

쉘 특수 문자 사용 예제

- - 이전 작업 디렉토리
  - `$ cd -`
- ‘’, “” 문자열 묶기
  - `$ echo ‘$SHELL’`
  - `$ echo “$SHELL”`
- `` back quotation
  - `$ echo “Today: `date`”`

쉘 특수 문자 – 파이프 (|)

- $ 명령 1 | 명령 2 | 명령 3 | …
  - 한 명령의 실행결과를 다음 명령의 입력으로 전달
  - 파이프 양쪽에 명령이 와야 함
  - `$ ls /etc | more`

입출력 방향 변경

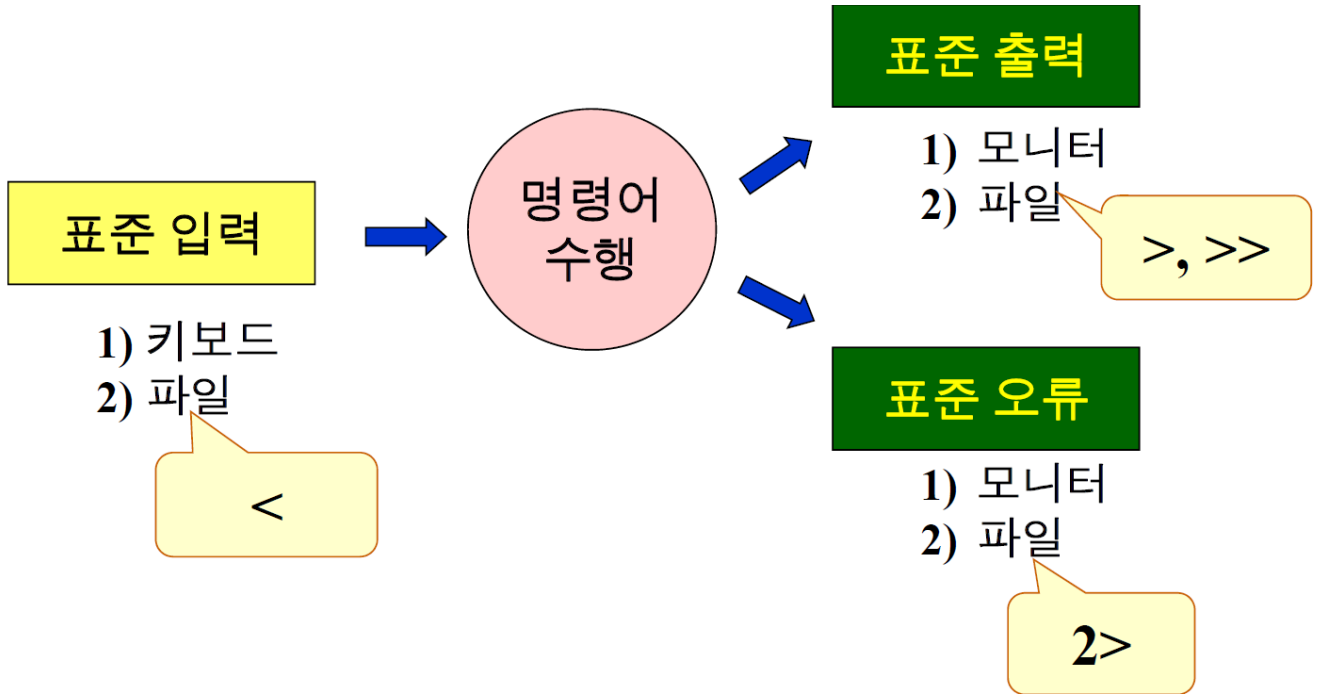
- 표준 입력 (Standard input)
  - 프로그램 실행에 필요한 데이터를 읽는 기본 장치
  - 기본 표준 입력: 키보드

- 표준 출력 (Standard output)
  - 프로그램의 실행 결과를 출력하는 장치
  - 기본 표준 출력: 모니터

- 표준 오류 (Standard error)
  - 프로그램 실행 중 발생한 오류 메시지를 출력하는 장치
  - 기본 표준 오류: 모니터

입출력 방향 변경

- 표준 입출력 장치를 변경 시 특수 기호 사용
- 리다이렉션 (Redirection): 표준 입출력 파일의 변경



출력 리다이렉션

- $ 명령 > 파일명　　또는　　$ 명령 >> 파일명
  - 표준출력을 모니터에서 파일로 변경
    - > : 새로운 파일로 생성, 기존 파일의 내용은 없어짐 (overwriting)
    - >> : 기존 파일의 끝에 내용 추가

```
$ ls -al > test
$ date > test
$ cat test
$ pwd >> test
$ cat test
```

  - ➔ test파일의 내용은?

출력 리다이렉션 실습

- 다음의 결과는?

```
1) $ ls -al
2) $ ls -al > ls.out
3) $ cat ls.out
4) $ pwd > ls.out
5) $ cat ls.out
6) $ date >> ls.out
7) $ cat ls.out
```

오류 리다이렉션

- $ 명령 2> 파일명
  - 표준 오류 메시지를 파일에 저장

- `$ ls /ttt`
  - ➔ 출력) 해당 파일이나 디렉토리가 없음

- `$ ls /ttt 2> ls.err`

- `$ cat ls.err`
  - ➔ 출력) 해당 파일이나 디렉토리가 없음

오류 리다이렉션 실습

```
1) $ ls /test
2) $ ls /test 2> ls.err
3) $ cat ls.err
4) $ rm ls.out ls.err
5) $ ls /var /test 1> ls.out 2> ls.err
6) $ cat ls.out
7) $ cat ls.err
8) $ rm ls.out ls.err
9) $ ls /var /test 1> ls.out 2>&1
10) $ cat ls.out
```

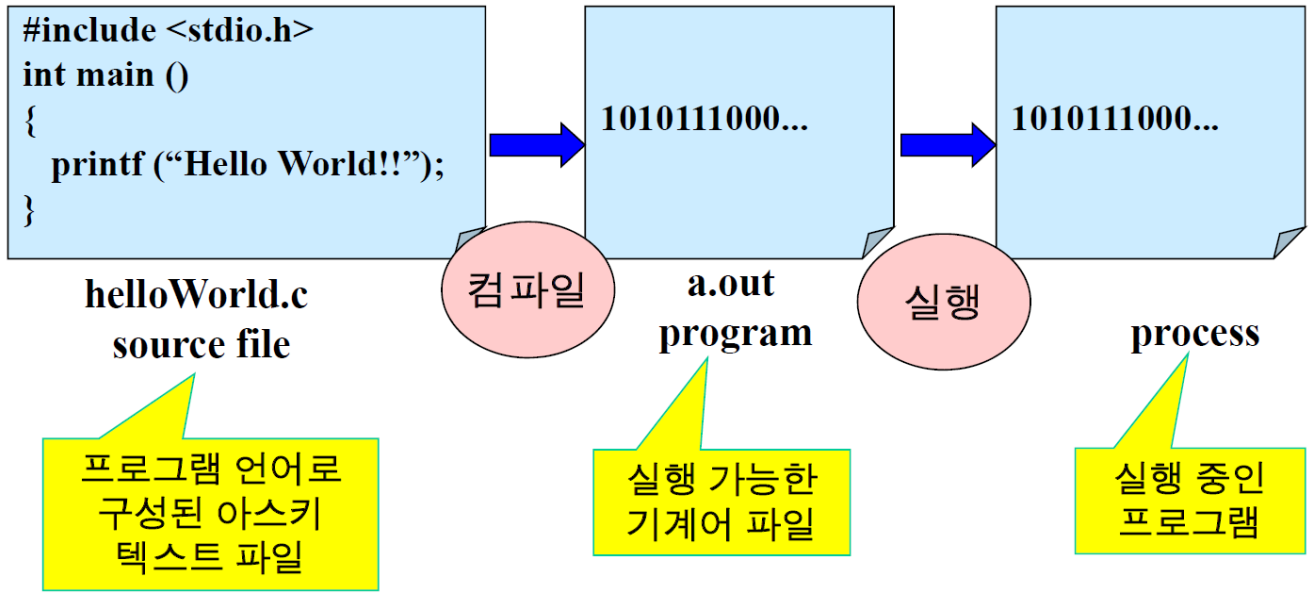
입력 리다이렉션

- $ 명령 < 파일명
  - 표준 입력을 키보드에서 파일로 변경
  - `$ cat < loveletter`

2. 프로세스 명령어와 사용자 명령어

프로세스

- 프로세스 (Process)
  - 현재 시스템에서 실행 중인 프로그램
  - 프로세스는 고유 번호를 가짐
    - Process ID : PID
    - 1번 프로세스 : init



프로세스

- Linux 프로세스의 종류

종류	설명
데몬 (daemon)	UNIX커널에 의해 시작되는 프로세스로 서비스 제공을 위한 프로세스들이다.
부모 (parent)	자식 프로세스를 만드는 프로세스
자식 (child)	부모에 의해 생성된 프로세스로 실행이 끝나면 부모 프로세스로 돌아간다.
고아 (orphan)	자식프로세스가 종료하기 전에 부모가 종료된 프로세스. 고아프로세스는 1번 프로세스를 새로운 부모로 가진다.
좀비 (zombie)	부모프로세스가 종료처리를 하지 않은 프로세스 프로세스테이블만 차지하고 있다.

프로세스 관리

- 프로세스 목록 보기
  - ps, pgrep

- 프로세스 종료
  - kill, pkill

- Foreground (전위)와 background (후위) 작업 제어
  - fg, bg, jobs

프로세스 목록 보기

- ps [옵션]
  - process status
  - 프로세스 정보를 출력
    - PID, 터미널, CPU 시간 등
  - 옵션
    - -e: 시스템에 있는 모든 프로세스 목록을 출력
    - -f: 프로세스에 대한 자세한 정보 출력
    - -u uid: 특정 사용자에 속한 모든 프로세스 출력
  - `$ ps -ef | more`

구분	설명	구분	설명
UID	소유자의 사용자 ID	STIME	프로세스 시작시간
PID	프로세스 번호	TTY	터미널 번호(? : 데몬)
PPID	부모 프로세스 번호	TIME	CPU 사용시간
C	프로세스 우선순위	CMD	명령어 이름

38. 다음은 ihduser 사용자가 로그인 후에 사용 중인 셸을 확인하는 과정이다. (괄호) 안에 들어갈 내용으로 알맞은 것은?

[ihduser@kait ~]$ (괄호)

❶ ps　② chsh -s
③ chsh -l　④ chsh -u

프로세스 목록 보기

- pgrep [옵션] 패턴
  - 프로세스 이름으로 찾아 정보를 출력
    - ‘ps [옵션] | grep 패턴’ 과 같은 기능
  - 옵션
    - -x: 패턴과 정확히 일치하는 PID 출력
    - -n: 패턴을 포함하고 있는 가장 최근의 PID 출력
    - -U uid: 특정 사용자에 속한 PID 출력
    - -l : PID와 프로세스 이름 출력
    - -t term: 특정 터미널과 관련된 프로세스 출력

```
$ pgrep telnet
$ ps | grep telnet
$ pgrep –n vi
$ pgrep -l telnet
$ pgrep –lt pts/2
```

프로세스 목록 보기 실습

```
1)$ ps
2)$ ps -f
3)$ ps -e
4)$ ps -e | more
5)$ ps -ef | more
6)$ ps -ef | grep 로그인 id
7)$ ps -u 로그인 id
```

프로세스 종료 시키기

- 프로세스의 종료
  - ps 명령으로 찾은 프로세스 중 불필요한 프로세스를 강제로 종료시키는 것
  - 프로세스를 종료시키면 그 자식 프로세스들도 같이 종료됨
  - 프로세스 종료시킬 때 PID나 프로세스 이름을 알아야 함

- 시그널 (Signal)
  - 프로세스에게 보내는 신호
  - 프로세스는 이 신호에 응답함 (무시 또는 종료 등)
  - kill, pkill 명령으로 신호를 보내는 것
  - (참고) $ man signal 또는 man –s 5 signal 로 자세한 정보 찾아볼 수 있음

프로세스 종료 시키기

- 시그널의 종류

시그널 번호	시그널 이름	기능	기본 응답
1	SIGHUP	• 터미널 연결이 끊어진 경우에 발생	종료
2	SIGINT	• 보통 Ctrl-C에 의해 발생	종료
9	SIGKILL	• 프로세스를 kill 시킨다. • 이 시그널은 무시할 수 없다.	종료
15	SIGTERM	• 프로세스를 종료시킨다. • 이 시그널은 무시할 수도 있다. • kill 명령이 보내는 기본 시그널	종료

프로세스 종료 시키기

- kill [시그널] pid 또는 pkill [시그널] 프로세스명
  - 지정한 프로세스들에게 시그널을 보냄
    - 사용자가 소유한 프로세스만 종료 가능
    - root 는 모든 프로세스를 종료시킬 수 있음
  - kill 명령은 default로 15번 (SIGTERM) 시그널을 보냄 (soft kill)
  - kill 명령을 사용하기 전에 대상 프로세스의 PID를 알고 있어야 함 (ps 또는 pgrep 활용)
  - 시그널
    - -9: 강제종료

25. 다음 명령의 결과에 대한 설명으로 알맞은 것은?

```
# kill 513
```

① PID가 513번인 프로세스에 1번 시그널을 전송한다.
② PID가 513번인 프로세스에 9번 시그널을 전송한다.
❸ PID가 513번인 프로세스에 15번 시그널을 전송한다.
④ kill 명령어는 프로세스명을 사용하므로 명령 오류가 발생한다.

프로세스 종료 시키기 실습

```
1) $ ps
2) $ sleep 100 &
3) $ ps
4) $ kill -9 PID(sleep)
5) $ vi /etc/hosts
```

- * 다른 터미널에서

```
1)ps
2)kill -9 PID(vi)
```

포그라운드와 백그라운드 프로세스

- Linux는 다중 작업을 지원하는 운영체제
  - Multi tasking
  - ➔ 동시에 여러 작업 수행 가능

23. 다음 제시된 명령을 백그라운드 프로세스로 실행하려고 할 때 (괄호) 안에 들어갈 내용으로 알맞은 것은?

```
# find / -name '*.txt' > list ( 괄호 )
```

① ;　② |

❸ &　④ +

- 포그라운드 처리 (전위처리)
  - 사용자가 명령을 입력한 후 결과가 출력될 때까지 기다려야 하는 경우 (`$ find / -name passwd`)
  - 보통의 명령처리 방법
- 백그라운드 처리 (후위처리)
  - 명령의 처리결과 출력과 관계없이 곧바로 프롬프트가 출력되어 다른 작업을 계속 할 수 있는 경우
  - 명령 실행 시 마지막에 & 를 붙임 (`$ find / -name passwd &`)

포그라운드와 백그라운드 작업제어

- 작업 (jobs)과 프로세스
  - job: 쉘이 관리할 수 있는 프로세스
  - 쉘은 job을 시작시키고 제어함
  - job은 프로세스이므로 각 job은 PID를 가지고 있음
  - 쉘이 할당한 일련번호인 job ID도 가지고 있음
  - 쉘은 동시에 여러 개의 job이 동작하도록 할 수 있음

포그라운드와 백그라운드 작업제어

- 작업 제어
  - 포그라운드 작업 → 포그라운드 프로세스
  - 백그라운드 작업 → 백그라운드 프로세스
  - 작업목록 보기
  - 작업 정지/종료/재동작

포그라운드와 백그라운드 작업제어

- jobs [%작업번호]
  - 백그라운드 작업을 모두 출력
  - 특정 작업번호를 지정할 경우 해당 작업의 정보만 출력
  - 작업번호
    - % 번호: 해당 작업의 작업 정보를 출력
    - %+ 또는 %%: 작업순서가 +인 작업 정보를 출력
    - %-: 작업 순서가 – 인 작업 정보를 출력
  - 사용법
    - `$ sleep 100 &`
    - `$ jobs`
    - `$ jobs %1`

21. 다음 중 백그라운드로 수행 중인 프로세스를 확인하는 명령어로 알맞은 것은?

① bg ② fg

❸ jobs ④ nohup

포그라운드와 백그라운드 작업제어

- jobs 명령 출력 항목

항목	출력예제	의 미
작업번호	[1]	작업번호로 백그라운드로 실행시킬 때마다 순차적으로 증가([1],[2],[3]...)
작업순서	+, -	작업순서를 표시 • + : 가장 최근에 접근한 작업 • - : + 작업보다 바로 전에 접근한 작업 • 공백 : 그 외의 작업
상태	실행중	작업의 상태를 표시 • 실행중(Running) : 현재 실행중 • 완료됨(Done) : 작업이 정상적으로 종료 • 종료됨(Terminated) : 작업이 비정상적으로 종료 • 정지(Stopped) : 작업이 잠시 중단됨.
명령	sleep 100&	실행중인 명령

포그라운드와 백그라운드 작업제어

- 작업전환 및 종료 명령

명령	기능
bg [%작업번호]	현재 작업이나 특정 작업을 백그라운드로 전환시켜 실행
fg [%작업번호]	현재 작업이나 특정 작업을 포그라운드로 전환시켜 실행
ctrl+z	포그라운드 작업을 중지시키고, 백그라운드의 중지된 목록으로 보냄
stop %작업번호	백그라운드에서 수행중인 특정 작업을 중지
kill %n	특정 작업을 종료

포그라운드와 백그라운드 작업제어

- 예제
  - `$ sleep 100`
  - `Ctrl + z` ← 포그라운드 작업을 중지시키고 백그라운드로 전환
  - `bg %1`
  - `jobs`
  - `fg`

포그라운드와 백그라운드 작업제어 실습

1) $ vi /etc/hosts
2) Ctrl+z
3) $ sleep 300&
4) $ jobs
5) $ kill %2
6) $ jobs
7) $ fg
8) $ vi 저장후 종료

- 1) $ sleep 150 &
- 2) $ sleep 200 &
- 3) $ jobs
- 4) $ fg %1
- 5) Ctrl + Z
- 6) $ jobs
- 7) $ stop %2
- 8) $ kill %1
- 9) $ kill %2
- 10) $ jobs

작업제어 명령

- nohup 백그라운드 명령
  - no hang up (끊지마!)의 약자
  - 로그아웃한 다음에도 백그라운드 작업은 작업이 완료될 때까지 실행하도록 해야할 때 nohup 사용
    - 백그라운드 작업을 실행시킨 단말기가 종료되거나 사용자가 로그아웃하면
      실행 중이던 백그라운드 작업은 함께 종료됨
  - 명령의 실행결과와 오류메시지는 현재 디렉토리에 nohup.out파일로 자동저장
  - `$ nohup find / -name passwd &`

  - (참고) nohup으로 실행시킨 파일은 반드시 755 퍼미션을 가지고 있어야 함

24. 다음 중 작업 중인 터미널이 닫혀야 실행 중인 프로세스를 계속해서 백그라운드 프로세스로 유지하려고 할 때 사용하는 명령어로 알맞은 것은?

① bg ② fg

③ jods ❹ nohup

사용자 정보

- 로그인한 사용자 정보 보기
  - users, who, w
- 사용자 자기자신의 정보 보기
  - who am i, whoami, id

사용자명 출력

- who
  - 시스템을 사용하고 있는 사용자의 정보를 출력
  - 옵션
    - -q: 사용자명만 출력
    - -H: 출력항목의 제목도 함께 출력
    - -b: 마지막으로 재부팅한 날짜와 시간을 출력

사용자명 출력

- w [사용자명]
  - 로그인한 사용자정보와 현재 하고 있는 작업 정보를 출력

- who am i
  - who 명령의 결과 중 자기 자신에 대한 정보만 출력

- whoami
  - 사용자의 로그인 ID를 출력

사용자명 출력

- id [옵션]
  - 사용자의 로그인 ID와 그룹 정보를 출력
  - 옵션
    - -a: 기본 그룹 외에 2차 그룹 정보도 출력

```
1) $ ls -al
2) $ ls -al > ls.out
3) $ cat ls.out
4) $ pwd > ls.out
5) $ cat ls.out
6) $ date >> ls.out
7) $ cat ls.out
```

1) who
2) who –m
3) who –q
4) who –H
5) w
6) who am I
7) whoami
8) id
9) id -a

3. 리눅스 유틸리티

Linux 유틸리티의 개념

- Linux 유틸리티
  - Linux 시스템 내에서 시스템 관리, 파일 처리, 네트워킹 및 보안 관리 등 다양한 작업을 수행
  - ➔ Bash, grep, tar, curl, wget, apt-get, crontab, netstat, ifconfig, sort, split, etc.

파일 정보 수집

- wc [옵션] 파일
  - 파일의 라인 수, 단어 수, 바이트, 문자 수 등 출력
  - 옵션
    - -c : 바이트 수
    - -m : 문자 수
    - -C: -m과 동일
    - -l : 라인 수
    - -w: 화이트 스페이스나 새로운 행으로 구분된 단어 수

파일 정보 수집 실습

- 결과는?

```
$ ls -l > test_wc
$ wc -c test_wc
$ wc -m test_wc
$ wc -cl test_wc
$ wc -mw test_wc
$ who | wc -l
```

```
1) $ wc /etc/passwd
2) $ wc /etc/hosts
3) $ wc -l /etc/services
4) $ cat /etc/hosts | wc -l
5) $ ls -l /usr/bin | wc -l
```

파일 정렬

- sort [옵션] 파일
  - 아스키 코드값을 기준으로 파일의 텍스트 내용을 정렬하여 화면에 출력
  - 옵션
    - -b : 앞에 붙는 공백 무시
    - -c : 정렬이 되지 않은 상태로 출력
    - -d : 사전식 순서로 정렬. 숫자, 문자, 공백만 비교
    - -f : 대소문자 구분 안함
    - -m : 정렬된 파일을 통합
    - -n : 숫자를 산술 값으로 전환해 정렬
    - -r : 역순 정렬
    - -t 문자 : 지정한 문자를 필드 구분자로 사용
    - +번호 : 번호+1 필드를 기준으로 정렬
    - +pos1 -pos2:정렬하고자 하는 필드의 열을 지정, pos1부터 pos2열까지 정렬

파일 정렬

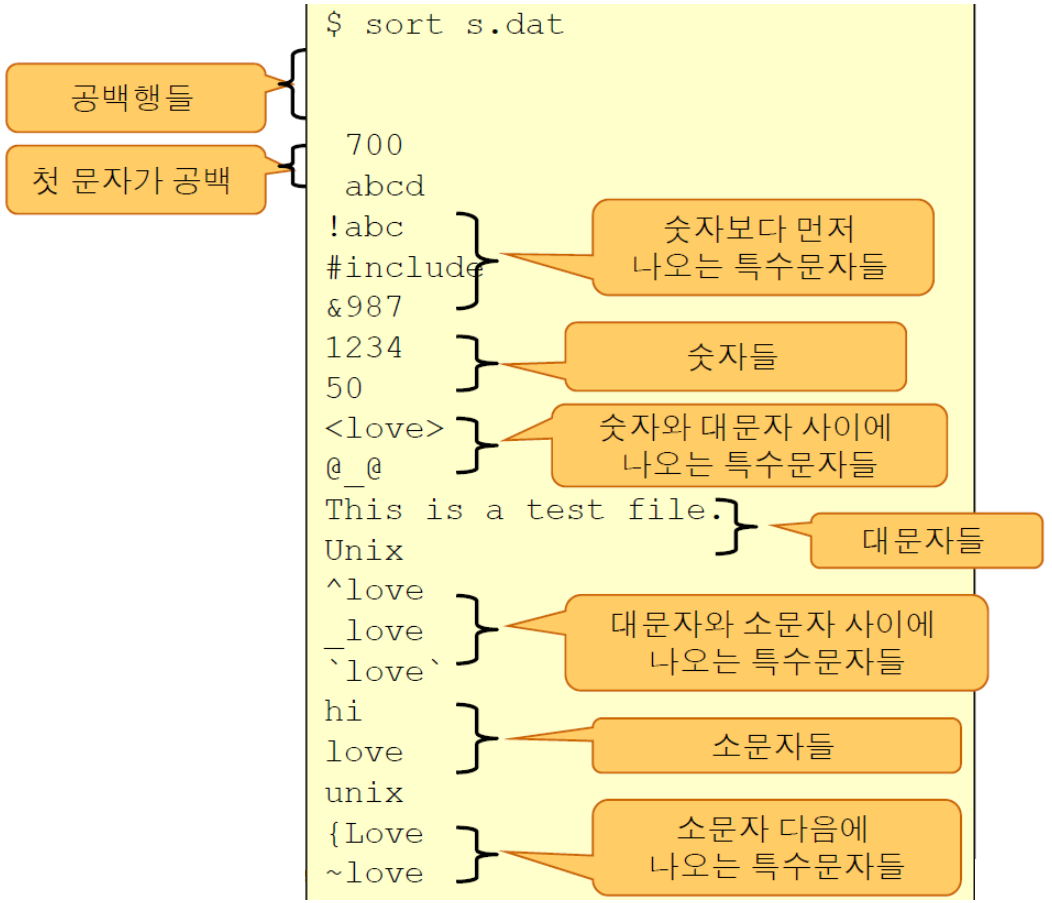
- 사용 방법
  - 기본 데이터 (s.dat)

```
$ more test_sort1
This is a test file.
1234

unix
Unix
_love
~love
!abc
  700
50
@_@
#include
love
^love

<love>
`love`
hi
&987
  abcd
{Love
$
```

  - 기본 정렬



파일 정렬

- 사용 방법
  - 사전식으로 정렬

```
$ sort -d test_sort1

@_@
 700
 abcd
1234
50
&987
{Love
This is a test file.
Unix
!abc
hi
#include
<love>
^love
_love
`love`
love
~love
unix
$
```

공백행들

첫문자가 공백문자

&는 무시하고 987로 간주

{ 는 무시하고 Love로 간주

!는 무시

앞의 특수문자들 무시

  - 역순으로 정렬

```
$ sort -r test_sort1
~love
{Love
unix
love
hi
`love`
_love
^love
Unix
This is a test file.
@_@
<love>
50
1234
&987
#include
!abc
 abcd
 700

$
```

파일 정렬

- 사용 방법
  - 필드 정렬

test_sort2.txt

001 Hong Gil-Dong 80 M

002 Park Ji-Soo 100 M

003 Lee Na-Young 54 F

004 Kim Chan-Sook 60 F

005 Han Ju-Hyun 75 M

006 Jyun Doo-Ri 49 F

007 Lee Mi-Ra 59 F

```
$ sort +1 test_sort2
005 Han Ju-Hyun 75 M
001 Hong Gil-Dong 80 M
006 Jyun Doo-Ri 49 F
004 Kim Chan-Sook 60 F
007 Lee Mi-Ra 59 F
003 Lee Na-Young 54 F
002 Park Ji-Soo 100 M
```

```
$ sort -k 2 test_sort2
005 Han Ju-Hyun 75 M
001 Hong Gil-Dong 80 M
006 Jyun Doo-Ri 49 F
004 Kim Chan-Sook 60 F
007 Lee Mi-Ra 59 F
003 Lee Na-Young 54 F
002 Park Ji-Soo 100 M
```

파일 정렬

- 사용 방법
  - 특정 필드를 기준으로 정렬

```
$ sort +1 -2 test_sort2
005 Han Ju-Hyun 75 M
001 Hong Gil-Dong 80 M
006 Jyun Doo-Ri 49 F
004 Kim Chan-Sook 60 F
003 Lee Na-Young 54 F
007 Lee Mi-Ra 59 F
002 Park Ji-Soo 100 M
```

2번 필드만 기준으로 정렬하고, 필드의 값이 같으면 3번 필드를 기준으로 정렬

파일 정렬

- 사용 방법
  - 숫자 기준으로 정렬 시 주의사항

```
$ sort -k 4 test_sort2
002 Park Ji-Soo 100 M
006 Jyun Doo-Ri 49 F
003 Lee Na-Young 54 F
007 Lee Mi-Ra 59 F
004 Kim Chan-Sook 60 F
005 Han Ju-Hyun 75 M
001 Hong Gil-Dong 80 M
```

```
$ sort -n -k 4 test_sort2
006 Jyun Doo-Ri 49 F
003 Lee Na-Young 54 F
007 Lee Mi-Ra 59 F
004 Kim Chan-Sook 60 F
005 Han Ju-Hyun 75 M
001 Hong Gil-Dong 80 M
002 Park Ji-Soo 100 M
```

파일 정렬

- 사용 방법
  - 정렬결과 저장

```
$ sort -n +3 -4 -o sort.out test_sort2
$ cat sort.out
006 Jyun Doo-Ri 49 F
003 Lee Na-Young 54 F
007 Lee Mi-Ra 59 F
004 Kim Chan-Sook 60 F
005 Han Ju-Hyun 75 M
001 Hong Gil-Dong 80 M
002 Park Ji-Soo 100 M
```

파일 정렬

- 사용 방법
  - 필드 구분자 (delimiter) 지정 –t:
    - `$ sort -t: /etc/passwd`

파일 분할

- split [옵션] [파일]
  - 큰 파일을 일정한 크기의 여러 개의 작은 파일로 분할
  - 분할된 파일 이름은 xaa, xab, … 순으로 생성
  - 옵션
    - -b n: 크기가 n바이트인 파일로 분할
    - -n: n줄씩 분할
    - 옵션을 지정하지 않으면 1000줄씩 분할
  - 파일을 지정하지 않으면 표준입력 내용을 분할 저장

파일 분할

- 사용 방법
  - 대상 데이터 파일 생성
    - `$ cp /etc/services  test_split`
    - `$ wc -l test_split`
  - 행 기준 분할
    - `$ split -30 test_split`
  - 바이트 기준 분할
    - `$ split -b 512 test_split`

중복 삭제

- uniq [옵션] [입력파일]
  - 파일 또는 표준입력으로 입력된 내용 중 중복된 내용의 줄이 연속으로 있으면 하나만 남기고 삭제
  - 파일을 지정하지 않으면 표준입력내용을 처리
  - 입출력 파일이름은 달라야 함
  - 옵션
    - -u : 중복되지 않는 줄만 출력
    - -d : 중복된 줄 중 1줄만 출력

중복 삭제

- 사용 방법

```
$ cat test_uniq1
aaaaa
abcde
abcde
bbbbb
abcde
bbbbb
bbbbb
abcde
ccc
```

- `$ uniq test_uniq1`

중복 삭제

- 사용 방법
  - 정렬과 중복제거를 동시에 수행
    - `$ sort test_uniq1 | uniq`
  - 중복없는 행 보기
    - `$ uniq -u test_uniq1`
  - 중복행과 중복 횟수 보기
    - `$ uniq -d test_uniq1`
    - `$ uniq -c test_uniq1`

필드 잘라내기

- cut [옵션] [파일]
  - 파일의 각 행에서 선택된 필드를 잘라냄
  - 옵션
    - -c 리스트: 각 줄에서 잘라낼 문자 위치 지정
    - -f 필드 수: 지정한 필드 잘라냄
    - -d 문자: 필드 구분자
      - 단독으로 쓰이지는 않음

필드 잘라내기

- 사용 방법

```
$ cat test_cut
001 Hong Gil-Dong 80 M
002 Park Ji-Soo 100 M
003 Lee Na-Young 54 F
004 Kim Chan-Sook 60 F
005 Han Ju-Hyun 75 M
006 Jyun Doo-Ri 49 F
007 Lee Mi-Ra 59 F
```

- 문자추출
  - `$ cut -c 1-2 test_cut`

필드 잘라내기

- 사용 방법
  - 필드 구분자 지정과 필드 추출
    - `$ cut -d ' ' -f 2 test_cut`
    - `$ cut -d: -f 1 /etc/passwd | more`

두 파일 연결하기

- paste [옵션] [파일1, 파일2, …]
  - 사용자가 지정한 두 개 이상의 파일 내용 중 같은 줄을 붙이거나 한 파일의 끝에 다른 파일의 내용 추가
  - 옵션
    - -s: 파일의 끝에 추가 (split으로 나눈 파일을 원래대로 붙일 때)
    - -d 문자: 필드 구분자
    - - : 파일 대신 표준 입력 사용

두 파일 연결하기

- 사용방법
  - 대상 파일 생성 (test_paste1, test_paste2, test_paste3)

```
001
002
003
004
005
006
007
```

```
Hong Gil-Dong
Park Ji-Soo
Lee Na-Young
Kim Chan-Sook
Han Ju-Hyun
Jyun Doo-Ri
Lee Mi-Ra
```

```
001 002 003 004
005 006 007
```

  - 파일 붙이기
    - `$ paste test_paste2 test_paste1`

두 파일 연결하기

- 사용방법
  - 두 파일의 행이 같지 않을 때
    - `$ paste test_paste2 test_paste3`
  - 필드 구분자 지정
    - `$ paste -d: test_paste1 test_paste2`
  - 파일 수평 붙이기
    - `$ paste -s test_paste1 test_paste2`

두 파일 연결하기

- cut과 paste의 복합 사용
  - 대상 파일 생성 (s.dat, u.dat)

```
Kim Ji-Soo    10 F
Lee Gil-Dong  20 M
Lee Gil-San   15 M
Park Ji-Soo   21 F
Choi Na-Na    20 F
```

```
aaaaa
abcde
abcde
bbbbb
```

  - `$ cut -d ' ' -f 1 s.dat | paste - u.dat`

파일 덤프

- dd [옵션] [if=입력파일] [of=출력파일]
  - 지정한 입력 파일을 지정한 옵션에 따라 변환하여 출력파일로 저장하는 유틸리티
  - 옵션
    - bs=n: 입출력 블록의 크기를 n바이트로 지정
    - conv=lcase: 알파벳을 소문자로 변환
    - conv=ucase: 알파벳을 대문자로 변환

파일 덤프

- 사용 방법
  - 대소문자 전환하기
    - `$ dd conv=lcase if=test_cut of=test_dd1`
  - 파일 지우기
    - `$ dd if=/dev/null of=test_dd1`

파일 덤프 실습

```
1) $ mkdir Practice
2) $ cd Practice
3) $ pwd
4) $ more /etc/passwd
5) $ cut -f 1 -d: /etc/passwd > login_list
6) $ cut -f 5 -d: /etc/passwd > name_list
7) $ paste -d: name_list login_list > user_list
8) $ sort -o user_list user_list
```

기타 유용한 유틸리티

- crontab
  - 시간 기반 작업 스케줄러입
  - 사용자는 crontab을 이용해 반복적인 작업을 자동으로 실행할 수 있음
  - crontab 기본 구조
    - 각 라인은 하나의 작업을 정의
    - 각 라인은 다섯 개의 시간 필드와 실행할 명령어로 구성
    - 분(0-59) 시(0-23) 일(1-31) 월(1-12) 요일(0-7)
  - ex)
    - * * * * * ls –al ← 매분마다 실행
    - * * * * * python alarm.py

기타 유용한 유틸리티

- crontab
  - 시간 필드 구성 예시
    - `$ 0 0 * * * /home/user/daily_job.sh` ← 매일 자정에 실행
    - `$ 0 8 * * 1 /scripts/weekly_update` ← 매주 월요일 오전 8시에 실행
    - `$ 0 2 1,15 * * /usr/local/bin/generate-report` ← 매월 1일과 15일에 실행
    - `$ 30 * * * * /path/to/job` ← 매 시간 30분에 실행
  - 특별 문자
    - * : 모든 값
    - , : 값 나열 (1, 3, 5 등)
    - - : 범위 지정 (1-5 등)
    - / : 단계 지정 (*/2 ← 매 시간마다)

기타 유용한 유틸리티

- crontab
  - crontab 관리 명령어
    - crontab -e: 현재 사용자의 crontab 파일을 편집
    - crontab -l: 현재 사용자의 crontab 작업 목록을 표시
    - crontab -r: 현재 사용자의 crontab 작업을 삭제

29. 다음 중 cron을 이용해서 매주 1회만 작업 스크립트를 실행하려고 할 때 (괄호) 안에 들어갈 내용을 알맞은 것은?

(괄호) /etc/work.sh

① 4 0 * 1 *　　② 4 0 1 * *

❸ 4 0 * * 2　　④ 4 0 * 2 *

기타 유용한 유틸리티

- apt-get
  - Debian 계열 Linux에서 소프트웨어 패키지를 관리하기 위한 유틸리티
  - Redhat 계열(Fedora, CentOS 등) 에서는 yum 또는 zypper

13. 다음 중 온라인 기반 패키지 관리 도구로 거리가 먼 것은?

① apt-get ② yum

③ zypper ❹ YaST

  - 주요 패키지 목록 업데이트
    - `$ sudo apt-get update`
  - 패키지 설치
    - `$ sudo apt-get install [패키지명]`
  - 패키지 제거
    - `$ sudo apt-get remove [패키지명]`
    - `$ sudo apt-get purse [패키지명]` ← 패키지 제거 및 구성 파일까지 모두 제거

기타 유용한 유틸리티

- apt-get
  - 의존성 문제 해결
    - `$ sudo apt-get -f install [패키지명]`
  - 자동 제거
    - `$ sudo apt-get autoremove`
  - 패키지 검색
    - `$ apt-cache search [검색어]`
      - 사용 가능한 패키지 목록에서 특정 패키지 검색
    - `$ apt list --installed`
    - `$ dpkg --list`

Assignment 02

- LMS 참조

End of slide