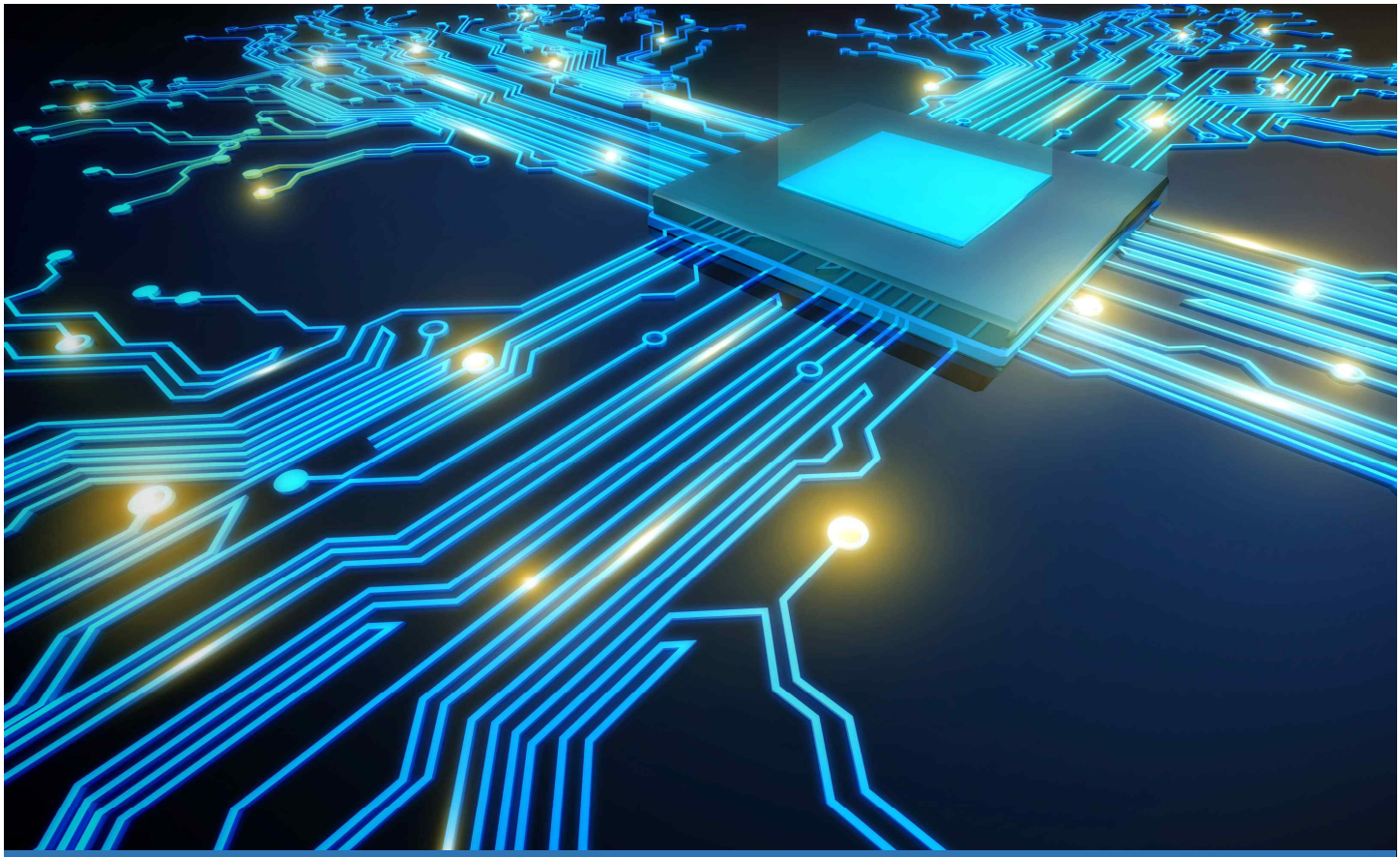
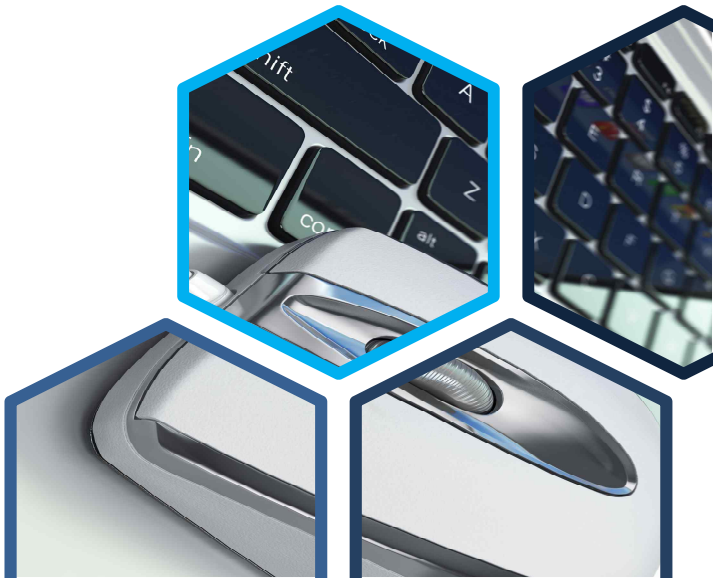


컴퓨터시스템

셸 활용하기



셸 활용하기

학습목표

- 셸의 기능과 종류를 설명할 수 있다.
- 셸의 기본 사용법을 설명할 수 있다.
- 입출력 방향 바꾸기를 실행할 수 있다.

학습내용

- 셸의 기능과 종류
- 셸의 기본 사용법
- 입출력 방향 바꾸기

셸의 기능과 종류

1 셸의 기능

명령어 해석기 기능

프로그래밍 기능

사용자 환경 설정 기능

명령어 해석기 기능

- 사용자와 커널 사이에서 명령을 해석하여 전달하는 해석기(Interpreter)와 번역기(Translator) 기능
- 사용자가 로그인하면 셸이 자동으로 실행되어 사용자가 명령을 입력하기를 기다림 → 로그인 셸

로그인 셸	/etc/passwd 파일에 사용자별로 지정

- 프롬프트: 셸이 사용자의 명령을 기다리고 있음을 나타내는 표시

셸의 기능과 종류

1 셸의 기능

프로그래밍 기능

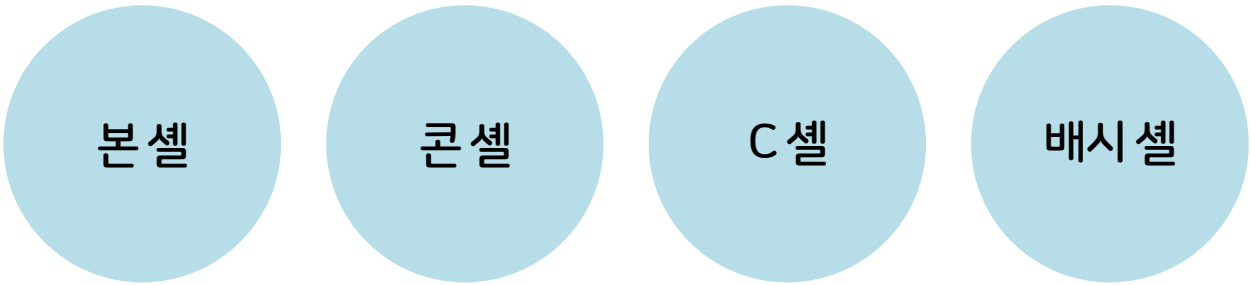
- 셸은 자체 내에 프로그래밍 기능이 있어 반복적으로 수행하는 작업을 하나의 프로그램으로 작성 가능
- 이러한 셸 프로그램을 셸 스크립트라고 지칭함

사용자 환경 설정 기능

- 사용자 환경을 설정할 수 있도록 초기화 파일 기능 제공
- 초기화 파일 기능
  - 명령을 찾아오는 경로 설정
  - 파일과 디렉터리를 새로 생성할 때 기본 권한을 설정함
  - 다양한 환경 변수 등을 설정함

셸의 기능과 종류

2 셸의 종류



본 셸(Bourne shell)

- 유닉스 V7에 처음 등장한 최초의 셸
- 개발자의 이름인 스티븐 본(Stephen Bourne)의 이름을 따서 본 셸이라고 함
- 초기에 단순하고 처리 속도가 빨라서 많이 사용
- 경로를 확인해보면 배시 셸과 심벌릭 링크로 연결되어 있음
- 지금도 시스템 관리 작업을 수행하는 많은 셸 스크립트는 본 셸을 기반으로 함
- 히스토리, 에일리어스, 작업 제어 등 사용자의 편의를 위한 기능을 제공하지 못해 이후에 다른 셸들이 등장
- 본 셸의 경로를 확인해보면 배시 셸과 심벌릭 링크로 연결되어 있음

```
[user1@localhost ~]$ ls -l /bin/sh
lrwxrwxrwx. 1 root root 4 5월 30 13:53 /bin/sh -> bash
[user1@localhost ~]$
```

셸의 기능과 종류

2 셸의 종류

C 셸(C shell)

- 캘리포니아대학교(버클리 캠퍼스)의 빌 조이(Bill Joy)가 개발
- 2BSD 유닉스에 포함되어 발표
- 본 셸에는 없던 에일리어스나 히스토리 같은 사용자 편의 기능 포함
- 셸 스크립트 작성을 위한 구문 형식이 C 언어와 같아 C 셸이라는 이름을 가지게 되었음
- C 셸의 명령 이름

```
csh
```

콘 셸(Korn shell)

- 1980년대 중반 AT&T 벨연구소의 데이비드 콘(David Korn)이 콘 셸을 개발
- 유닉스 SVR 4에 포함되어 발표
- C 셸과 달리 본 셸과의 호환성을 유지하고 히스토리, 에일리어스 기능 등 C 셸의 특징도 모두 제공하면서 처리 속도도 빠름
- 콘 셸의 명령 이름

```
ksh
```

셸의 기능과 종류

2 셸의 종류

배시 셸(bash shell)

- 본 셸을 기반으로 개발된 셸로서 1988년 브레인 폭스(Brain Fox)가 개발
- 본 셸과 호환성을 유지하면서 C 셸, 콘 셸의 편리한 기능도 포함
- 배시 셸의 명령 이름

```
bash
```

- 배시 셸의 모든 버전은 GPL 라이선스에 의거하여 자유롭게 사용 가능
- 리눅스의 기본 셸로 제공되고 있어 리눅스 셸로도 많이 알려짐

셸의 기본 사용법

1 기본 셸 바꾸기

셸 종류 알아보기

- 프롬프트 모양 참조

$	본 셸, 배시 셸, 콘 셸의 기본 프롬프트
%	C 셸의 기본 프롬프트

- 사용자 정보 확인: /etc/passwd 파일
  - 가장 앞에 나온 정보가 로그인 ID
  - 사용자 정보의 가장 마지막에 나온 /bin/bash가 기본 셸

```
[user1@localhost ~]$ grep user1 /etc/passwd
user1 :x:1000:1000:user1 :/home/user1 :/bin/bash
[user1@localhost ~]$
```

셸의 기본 사용법

1 기본 셸 바꾸기

기본 셸 바꾸기: chsh

- 기능: 사용자 로그인 셸을 바꿈
- 형식: chsh [옵션] [사용자명]
- 옵션

-s shell	지정한 셸(절대 경로)로 로그인 셸을 바꿈
-l(소문자 L)	/etc/shells 파일에 지정된 셸을 출력함

- 사용 예

```
chsh
chsh -l
chsh –s /bin/sh user1
```

- 바꿀 수 있는 셸의 종류: /etc/shells 파일에 지정
  - 본 셸, 대시 셸, 배시 셸이 사용 가능하다는 것을 표시

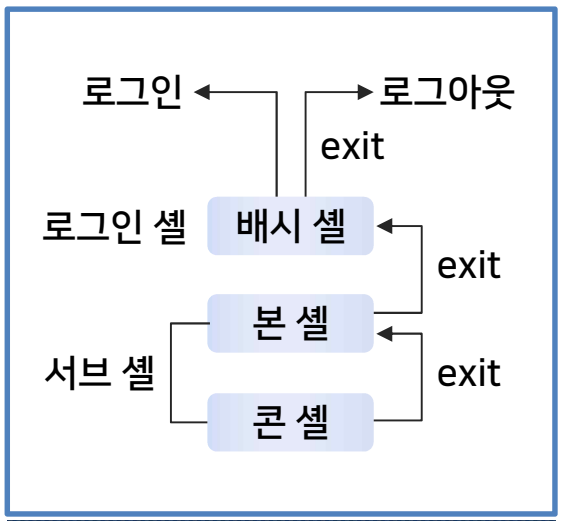
```
[user1@myubuntu:~$ cat /etc/shells
# /etc/shells: valid login shells
/bin/bash
/bin/rbash
/bin/dash
```

셸의 기본 사용법

1 기본 셸 바꾸기

로그인 셸과 서브 셸

- 프롬프트에서 다른 셸을 실행할 수 있는데 이를 서브 셸이라 함
- 서브 셸은 또 다른 서브 셸 생성 가능
- 서브 셸을 종료하는 명령: ^d(ctrl+d), exit 등
  - 서브 셸이 종료되면 서브 셸을 실행했던 이전 셸 환경으로 복귀
- 로그인 셸에서 로그아웃하면 접속 해제



로그인 셸과 서브 셸

셸 내장 명령

- 셸은 자체적으로 내장 명령을 가지고 있음
- 셸 내장 명령은 별도의 실행 파일이 없고 셸 안에 포함됨
- 셸 내장 명령의 예

```
cd
```

- 일반 명령(실행 파일)의 경우
  - 실행 파일은 바이너리 파일이므로 cat 명령으로 파일의 내용을 확인할 수 없음

셸의 기본 사용법

2 셸 내장 명령 및 출력 명령

배시 셸의 출력 명령: echo

- 기능: 화면에 한 줄의 문자열을 출력
- 형식: echo [옵션] [문자열]
- 옵션

-n	마지막에 줄바꿈을 하지 않음

- 사용 예

```
echo
echo text
echo –n text
```

배시 셸의 출력 명령: printf

- 기능: % 지시자와 ₩ 문자를 이용하여 출력 형식을 지정 가능
- 형식: printf [옵션] [인수]
- 옵션

%d, \ n 등	언어 printf 함수의 형식을 지정함

- 사용 예

```
printf text
printf “text \ n”
printf  “%d \ n” 100
```

셸의 기본 사용법

3 특수문자 사용하기

특수문자

- 사용자가 더욱 편리하게 명령을 입력하고 실행할 수 있도록 다양한 특수문자 제공

주요 특수문자	*, ?, |, ;, [ ], ~, ‘ ’, “ ”, ` `` 등

- 명령을 입력하면 셸은 먼저 특수문자가 있는지 확인하고 이를 적절한 형태로 변경한 후 명령을 실행

특수문자 *(별표)

- 임의의 문자열을 나타내는 특수문자로 0개 이상의 문자로 대체
- 사용 예

ls *	▶ **현재 디렉터리의 모든 파일과 서브 디렉터리를 나열함** ▶ **서브 디렉터리의 내용도 출력함**
cp * /tmp	▶ 현재 디렉터리의 모든 파일을 /tmp 디렉터리 아래로 복사함
ls -F s*	▶ s, smt, semt와 같이 파일명이 s로 시작하는 모든 파일의 이름과 파일 종류를 출력함 ▶ t도 해당한다는 데 주의해야 함
cp *.c ../ch3	▶ **확장자가 c인 모든 파일을 상위 디렉터리 아래의 ch3 디렉터리로 복사함**
ls -l p*t	▶ 파일명이 p로 시작하고 t로 끝나는 모든 파일의 상세 정보를 출력함 ▶ pt, pat, part, p12345t 등 이 조건에 맞는 모든 파일의 정보를 볼 수 있음

셸의 기본 사용법

3 특수문자 사용하기

특수문자 ?와 [ ]

- 하나의 문자를 나타내는 데 사용
- ?는 길이가 1인 임의의 한 문자를, [ ]는 괄호 안에 포함된 문자 중 하나를 나타냄
- 사용 예

ls s?.txt	▶ s 다음에 임의의 한 문자가 오고 파일의 확장자가 txt인 모든 파일의 이름을 출력함 ▶ s1.txt, s2.txt, sa.txt 등이 해당됨 (단, s.txt는 제외함)
ls -l smt[135].txt	▶ smt 다음에 1, 3, 5 중 하나가 오고 파일의 확장자가 txt인 모든 파일의 이름을 출력함 ▶ smt1.txt, smt3.txt, smt5.txt 파일이 있으면 해당 파일의 상세 정보를 출력함 (단, smt.txt는 제외함)
ls -l smt [1-3].txt	▶ [1-3]은 1부터 3까지의 범위를 의미함 → ls -l smt [123].txt와 결과가 같음 ▶ smt1.txt, smt2.txt, smt3.txt 파일이 있으면 해당 파일의 상세 정보를 출력함
ls [0-9]*	▶ 파일명이 숫자로 시작하는 모든 파일의 목록을 출력함
ls [A-Za-z]*[0-9]	▶ 파일명이 영문자로 시작하고 숫자로 끝나는 모든 파일의 목록을 출력함

셸의 기본 사용법

3 특수문자 사용하기

특수문자 ~와 -

- 사용 예

cp *.txt ~/ch3	▶ 확장자가 txt인 모든 파일을 현재 작업 중인 사용자의 홈 디렉터리 아래 ch3 디렉터리로 복사함
cp ~user2/linux.txt .	▶ User2라는 사용자의 홈 디렉터리 아래에서 linux.txt 파일을 찾아 현재 디렉터리로 복사함
cd -	▶ 이전 작업 디렉터리로 이동함

특수문자 ;과 |

- ;(쌍반점)과 |(파이프)는 명령과 명령을 연결

;	➡	연결된 명령을 왼쪽부터 차례로 실행
|	➡	왼쪽 명령의 실행 결과를 오른쪽 명령의 입력으로 전달

- 사용 예

date; ls; pwd	▶ 왼쪽부터 차례대로 명령을 실행함 ▶ 날짜를 출력한 후 현재 디렉터리의 파일 목록을 출력하고, 마지막으로 현재 작업 디렉터리의 절대 경로를 보여줌
ls -al / | more	▶ 루트 디렉터리에 있는 모든 파일의 상세정보를 한 화면씩 출력함 ▶ ls -al / 명령의 결과가 more 명령의 입력으로 전달되어 페이지 단위로 출력되는 것임

셸의 기본 사용법

3 특수문자 사용하기

특수문자 ' '와 " "

- ' '(작은따옴표)와 " "(큰따옴표)는 문자를 감싸서 문자열로 만들어주고, 문자열 안에 사용된 특수 문자의 기능을 없앰

' '	➡	모든 특수 문자를 일반 문자로 간주하여 처리
" "	➡	$, `, ₩을 제외한 모든 특수 문자를 일반 문자로 간주하여 처리

- 사용 예

echo '$SHELL'	▶ $SHELL 문자열이 화면에 출력됨
echo "$SHELL"	▶ 셸 환경 변수인 SHELL에 저장된 값인 현재 셸의 종류가 화면에 출력됨 예 ▶ /bin/bash

셸의 기본 사용법

3 특수문자 사용하기

특수문자 ` `

- 셸은 "` `"로 감싸인 문자열을 명령으로 해석하여 명령의 실행 결과로 전환
- 사용 예

echo "Today is `date`"	▶ `date`가 명령으로 해석되어 date 명령의 실행 결과로 바뀜 ▶ 결과적으로 다음과 같이 출력됨 Today is 2017.06.28. (수) 18:32:45 KST
ls /usr/bin/`uname -m`	▶ uname -m 명령의 실행 결과를 문자열로 바꾸어 파일 이름으로 사용함

특수문자 ₩

- ₩(역빗금, ₩와 동일함)은 특수문자 바로 앞에 사용하며, 해당 특수문자의 효과를 없애고 일반 문자처럼 처리
- 사용 예

ls -l t*	▶ t*라는 이름을 가진 파일의 상세 정보를 출력함 ▶ \ 없이 t*를 사용하면 t로 시작하는 모든 파일의 상세 정보를 출력함
echo \$SHELL	▶ $SHELL을 화면에 출력함 ▶ echo '$SHELL'과 결과가 같음

셸의 기본 사용법

3 특수문자 사용하기

특수문자 >, <, >>

- 입출력의 방향을 바꾸는 특수문자
- 사용 예

ls -l > res	▶ ls -l 명령의 실행 결과를 화면이 아닌 res 파일에 저장함

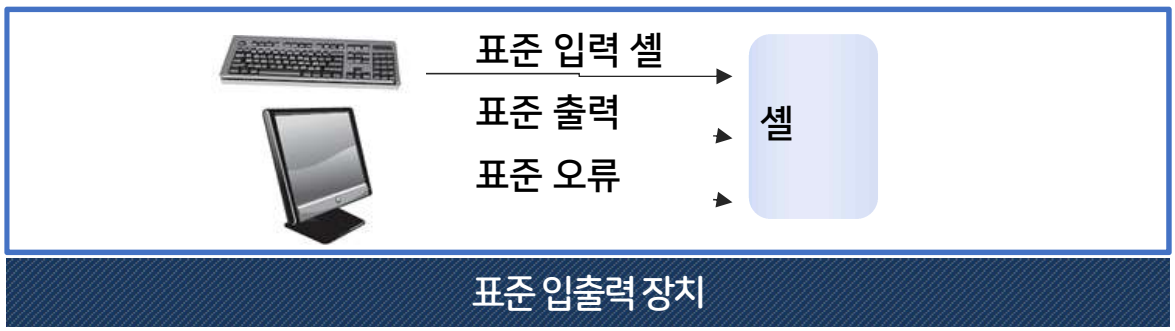
셸의 기본 사용법 실습 영상은
학습 콘텐츠에서 확인하실 수 있습니다.

입출력 방향 바꾸기

1 개요

표준 입출력 장치

표준 입력 장치	표준 출력 장치	표준 오류 장치
셸이 작업을 수행하는 데 필요한 정보를 받아들이는 장치 → 키보드	실행 결과를 내보내는 장치 → 모니터	오류 메시지를 내보내는 장치 → 모니터



입출력 방향 바꾸기

1 개요

파일

- 파일 관리를 위해 붙이는 일련 번호
- 입출력 장치를 변경할 때는 이 파일 디스크립터를 사용
- 표준 입출력 장치를 파일로 바꾸는 것을 '리다이렉션(Redirection)'이라고 함
- 표준 입출력 장치의 파일 디스크립터

파일 디스크립터	파일 디스크립터 대신 사용하는 이름	정의
0	stdin	명령의 표준 입력
1	stdout	명령의 표준 출력
2	stderr	명령의 표준 오류

방향 바꾸기

- 출력 재지정
- 입력 재지정
- >, >> (stdout을 파일로 저장 또는 추가)
- < (파일을 stdin으로 전달)

파이프(|)

- Process 연결 (stdout 을 stdin 으로 전달)
- Process 간의 데이터 흐름은 자동으로 조절됨

입출력 방향 바꾸기

2 출력 방향 바꾸기

출력 리다이렉션

>	→	기존 파일의 내용을 삭제하고 새로 결과를 저장
>>	→	기존 파일의 내용 뒤에 결과를 추가
1	→	파일 디스크립터 1번(표준 출력, 화면)

- 셸은 >를 사용한 리다이렉션에서 지정한 이름의 파일이 없으면 파일을 생성해서 명령의 수행 결과를 저장
- 파일이 있으면 이전의 내용이 없어지고 명령의 수행 결과로 대체
- 기능: 파일 리다이렉션(덮어쓰기)을 함
- 형식
  - 명령 1> 파일명
  - 명령 > 파일명

입출력 방향 바꾸기

2 출력 방향 바꾸기

예상치 않게 파일의 내용이 겹쳐 쓰이는 상황을 예방하기

- 설정하기

```
[user1@localhost ch4]$ set -o noclobber
[user1@localhost ch4]$ ls > outxx
bash: outxx: cannot overwrite existing file
[user1@localhost ch4]$
```

- 설정 해제

```
[user1@localhost ch4]$ set +o noclobber
[user1@localhost ch4]$ ls > outxx
[user1@localhost ch4]$
```

파일에 내용 추가하기: >>

- 기능: 파일에 내용을 추가함
- 형식: 명령 >> 파일명
- 지정한 파일 유·무

지정한 파일이 없는 경우	지정한 파일이 있는 경우
파일을 생성	기존 파일의 끝에 명령의 실행 결과 추가

입출력 방향 바꾸기

3 오류 방향 바꾸기

오류 리다이렉션

- 표준 오류도 기본적으로 화면으로 출력되며 표준 출력처럼 리다이렉션 가능

표준출력 리다이렉션	오류 메시지는 리다이렉션 안됨

- 오류 리다이렉션에서는 파일 디스크립터 번호를 생략 불가
- 기능: 표준 오류 메시지를 파일에 저장
- 형식: 명령 2> 파일명
- 표준 출력과 표준 오류를 한 번에 리다이렉션하기
  - 사용 예

```
[user1@localhost  ch4]$  ls  .  /abc > ls.out  2> ls.err
[user1@localhost  ch4]$
```

명령의 정상 실행 결과	→	파일로 리다이렉션(>).
그 명령 전체의 오류 메시지	→	번 파일(표준 출력 파일, &1이라고 표현함)로 리다이렉션(2>).

입출력 방향 바꾸기

4 입력 방향 바꾸기

입력 리다이렉션

- 기능: 표준 입력을 바꿈
- 형식
  - 명령 ⦿ < 파일명
  - 명령 < 파일명

입출력 방향 바꾸기 실습 영상은
학습 콘텐츠에서 확인하실 수 있습니다.

핵심요약

1 셸의 기능과 종류

- 셸의 기능: 명령어 해석기 기능, 프로그래밍 기능, 사용자 환경 설정 기능
- 셸의 종류: 본 셸, 콘 셸, C 셸, 배시 셸 등

2 셸의 기본 사용법

- 셸은 자체적으로 내장 명령을 가지고 있으며 별도의 실행 파일이 없고 셸 안에 포함되어 있음
- 일반 명령(실행 파일)의 경우 바이너리 파일이므로 cat 명령으로 파일의 내용을 확인할 수 없음
- 셸 프로그래밍의 두 가지 방법
  - 명령을 차례(Line Command)로 입력하고 Shell이 대화형으로 실행하는 방법
  - 하나의 스크립트 작성 후 프로그램처럼 사용하는 방법
- 셸 변수에 부여된 값은 echo 명령을 통해 확인 가능함

핵심요약

3 입출력 방향 바꾸기

- 표준 입력 장치: 셸이 작업을 수행하는 데 필요한 정보를 받아들이는 장치 → 키보드
- 표준 출력 장치: 실행 결과를 내보내는 장치 → 모니터
- 표준 오류 장치: 오류 메시지를 내보내는 장치 → 모니터