**명령 프롬프트(cmd.exe) 사용법**

**송 근 배**

TABLE OF CONTENTS

[1. 표준 입출력 6](#_Toc7551616)

[2. 리다이렉션 - Redirection 7](#_Toc7551617)

[2.1. 입력 리다이렉션 7](#_Toc7551618)

[2.2. 출력 리다이렉션 8](#_Toc7551619)

[2.3. 입출력 리다이렉션의 혼용 10](#_Toc7551620)

[2.4. 출력 리다이렉션의 다른 방향 10](#_Toc7551621)

[2.4.1. 파이프 - Pipe 11](#_Toc7551622)

[3. 참고 사이트 13](#_Toc7551623)

그림 목차

[[그림 1] 표준 입출력 7](#_Toc7549926)

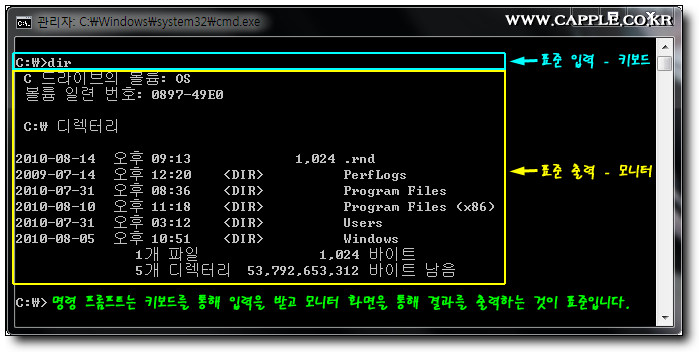
[[그림 1] 그림 샘플 17](#_Toc7549927)

표 목차

APPENDIX

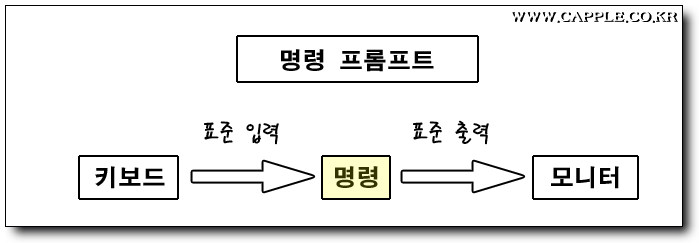
# 표준 입출력

윈도우의 명령어 쉘인 명령 프롬프트에는 **표준 입력**과 **표준 출력**이라는 것이 존재합니다. 명령 프롬프트는 기본적으로 표준 입력 장치는 **'키보드'** 로, 표준 출력 장치는 **'모니터'** 로 설정되어 있습니다. 즉, 이 말은 명령어의 입력은 키보드를 통해서 받고 명령의 처리 결과는 모니터 화면으로 출력해 주는 것이 표준이라는 것입니다.



[그림 1] 표준 입출력

이를 간단하게 표현하면 아래와 같습니다.

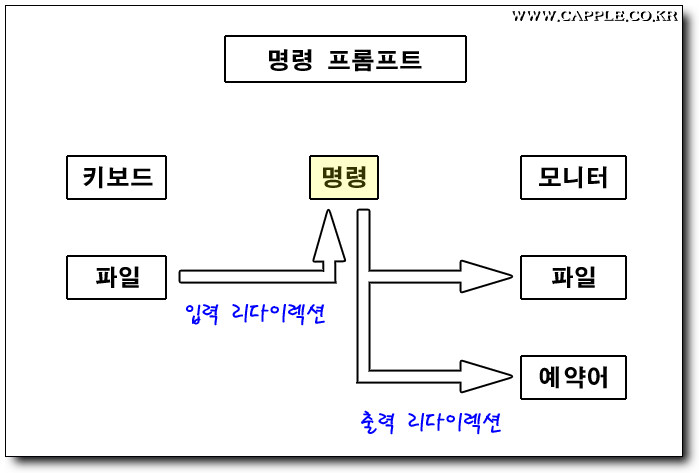


# 리다이렉션 - Redirection

명령 프롬프트의 표준 입출력을 키보드나 모니터가 아닌 파일이나 프린터와 같이 다른 방향으로 바꿀 수 있습니다. 이러한 것을 **리다이렉션(리디렉션, Redirection)** 과 **파이프(Pipe)** 라고 합니다.

※ 리다이렉션과 리디렉션은 같은 말입니다. 리다이렉션은 영국식 발음이며 리디렉션은 미국식 발음입니다. 과거엔 리다이렉션이라는 발음을 많이 사용했으나 점차 리디렉션이라는 발음을 많이 사용하는 편 입니다.

리다이렉션이란 사전적 의미로 다른 방식이나 용도로 전용하여 쓰다, 다른 주소나 방향으로 다시 보내다라는 의미를 가진 Redirect 의 명사형을 의미합니다. 이를 명령 프롬프트의 표준 입력과 표준 출력에 대입하면 표준 입력이나 표준 출력의 방향을 다른 곳에서 받거나 다른 곳으로 보낸다는 의미 정도가 될 수 있습니다. 명령 프롬프트에서 정의하는 리다이렉션의 의미도 이와 다르지 않습니다.

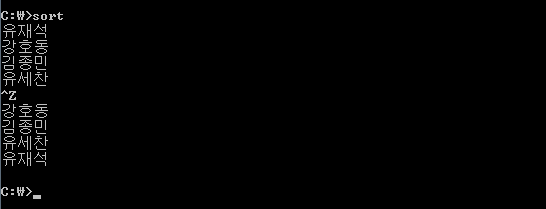


## 입력 리다이렉션

표준 입력의 리다이렉션은 다음의 의미를 가집니다.

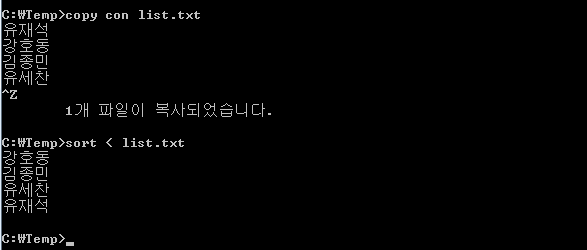
- 입력을 키보드에서 파일로 바꾸는 것  
- 리다이렉션을 의미하는 **<** 를 통해 입력 받을 파일을 지정 (명령 < 파일)

표준 입력의 리다이렉션을 설명하는데에는 sort 명령이 가장 많이 사용됩니다. sort 명령은 키보드로부터 입력된 데이터를 오름차순 또는 내림차순으로 정렬해 주는 명령입니다.



[그림 2] 입력 리다이렉션 예 (sort)

입력 리다이렉션은 이렇게 키보드로 입력받는 내용을 파일로부터 받는 것입니다. 다음 예제는 copy con 명령을 통해서 list.txt라는 파일을 만들고, 리다이렉션을 사용하여 해당 파일을 입력받아 sort 명령으로 정렬하는 것을 보여줍니다.



## 출력 리다이렉션

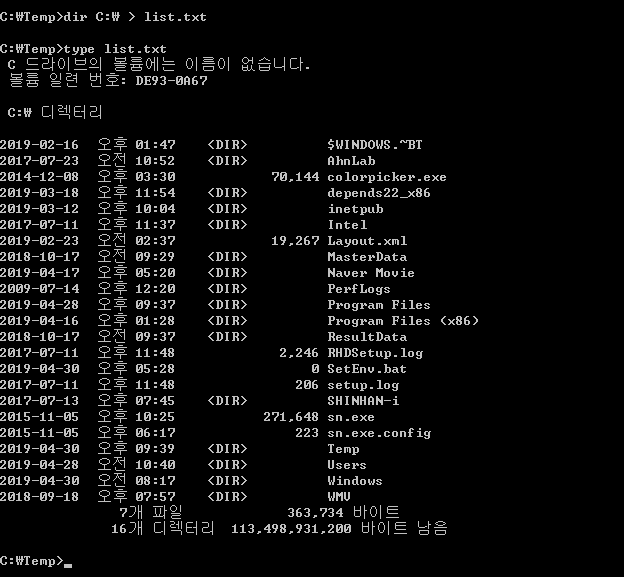
다음으로 표준 출력의 리다이렉션은 다음의 의미를 가집니다.

- 출력의 방향을 모니터에서 파일로 바꾸는 것

- 리다이렉션을 의미하는 **>** 또는 **>>** 를 통해 저장할 파일을 지정

> 는 기존에 파일이 있다면 무조건 새로 덮어쓰기 때문에 이를 원치 않고 기존의 파일에 추가하여 기록하고 싶다면 >> 을 쓰게 됩니다.

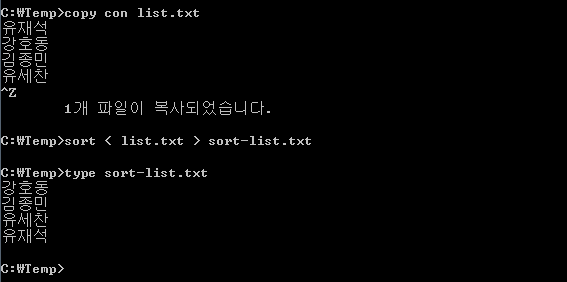
출력 리다이렉션은 화면으로 출력되는 내용을 파일로 저장을 하는 것입니다. 다음 예제는 리다이렉션을 통해 C: 드라이브의 파일 목록을 텍스트 파일로 저장하는 것을 보여줍니다.



[그림 3] 출력 리다이렉션 예

## 입출력 리다이렉션의 혼용

입력과 출력 리다이렉션은 목적에 따라 혼용하여 사용할 수 있습니다. 다음 예제는 list.txt 파일을 위에서 입력 리다이렉션을 사용하여 list.txt 라는 파일의 내용을 Sort 명령을 통해 오름차순으로 정렬한 후 출력했었죠? 그런데 여기에 출력 리다이렉션을 통해 정렬 결과를 화면으로 출력하지 않고 파일로 기록할 수도 있습니다. 즉, 입출력 리다이렉션이 모두 사용되는 것이죠.



[그림 4] 입출력 리다이렉션의 혼용 예

## 출력 리다이렉션의 다른 방향

위에서 이야기 했지만 출력 리다이렉션은 표준 출력의 방향을 다른 곳으로 향하게 하는 것 입니다. 위에서는 새로운 출력의 방향으로 파일을 예제로 들었지만 사실 파일로만 출력을 보낼 수 있는 것은 아닙니다. 윈도우에는 예약어라는 게 존재합니다. 대표적으로 CON, PRN, NUL, AUX 등이 있는데 이러한 예약어는 각각 윈도우에 의해 미리 지정된 장치를 의미하는 것으로 파일명이나 폴더명으로도 쓰지 못합니다. 대표적으로 CON은 키보드를 PRN 은 프린터를 NUL 은 값이 없는, 빈 곳을(null) 의미합니다.  
출력 리다이렉션을 통해 출력의 결과를 파일이 아닌 이러한 곳으로 보낼 수도 있습니다. 다른 것은 모르겠고 프린터를 의미하는 PRN 과 빈 곳을 의미하는 NUL 로는 가끔씩 보내기도 합니다.

* 명령의 결과를 프린터로 출력

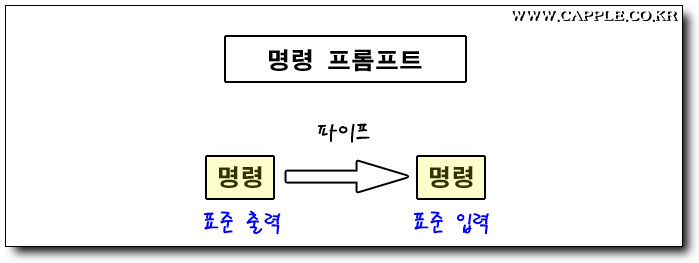
명령 > prn

* **명령의 결과를 출력하지 않음**

명령 > nul

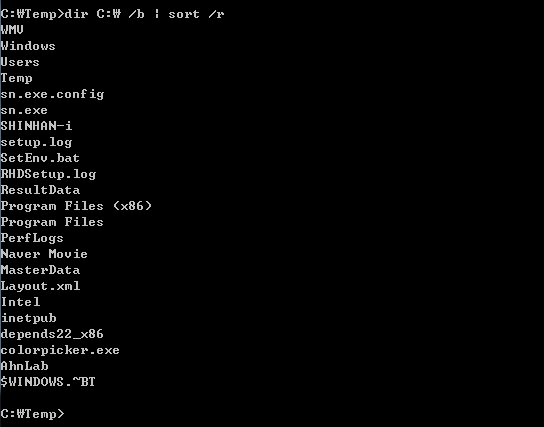
**파이프(Pipe)**

명령과 명령을 하나로 연결하여 앞선 명령의 출력을 뒤의 명령의 입력으로 연결해 줍니다. 즉, 출력이 입력으로 자연스럽게 흘러들어가는 겁니다.



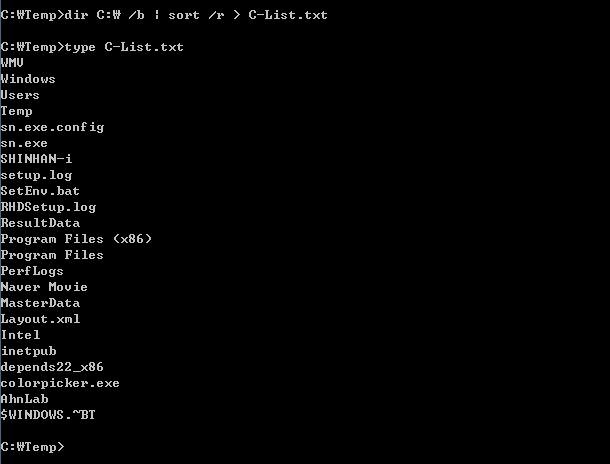
* 어떤 명령의 표준 출력을 다음 명령의 표준 입력으로 전환하여 두 명령을 마치 하나의명령처럼 만들어 주는 것
* 파이프를 의미하는 **|** 를 통해 명령과 명령을 연결
* **|** 를 기준으로 좌측 명령의 표준 출력이 우측 명령의 표준 입력이 됨. (표준 출력 | 표준 입력)

다음은 dir 명령과 sort 명령을 파이프를 통해 연결하여 간단하게 C: 드라이브의 파일 목록만을 역순으로 정렬하는 예제를 보여줍니다.



[그림 5] 파이프 사용 예

이러한 파이프는 리다이렉션과 조합하여 사용할 수도 있습니다.



# 명령어

## ren

윈도우 도스창 즉 명령 프롬프트 CMD 에서, 파일명을 변경하려면 ren 이라는 명령어를 사용합니다. ren은 rename 의 준말입니다. 즉 "이름 바꾸기"입니다. 예를 들어 현재 디렉토리(폴더)에 test.txt 라는 이름의 파일이 있을 때 다음과 같은 명령을 사용하면

**ren test.txt cool.txt**

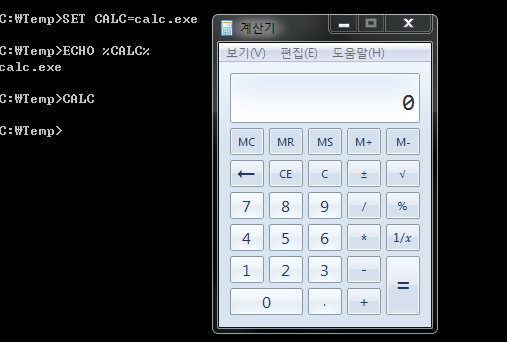
test.txt 라는 파일이 cool.txt 라는 새 이름으로 변경됩니다. 파일뿐만 아니라 "디렉토리(Directory)" 즉 폴더(Folder)도 마찬가지입니다. 예를 들어 000 이라는 하위 디렉토리 이름을 111 로 변경하려면 다음과 같이 합니다.

**ren 000 111**

# 환경 변수

## 환경 변수 설정 방법

SET 변수명=값



## 동적 환경 변수

* **%CD%** - 현재 디렉터리 문자열로 확장합니다.
* **%DATE%** - DATE 명령과 같은 포맷을 사용하여 현재 날짜로 확장합니다.
* **%TIME%** - TIME 명령과 같은 포맷을 사용하여 현재 시간으로 확장합니다.
* **%RANDOM%** - 0과 32767 사이에 있는 임의 숫자로 확장합니다.
* **%ERRORLEVEL%** - 현재 ERRPRLEVEL 값으로 확장합니다
* **%CMDEXTVERSION%** - 현재 명령 프로세서 확장자 버전 번호로 확장합니다.
* **%CMDCMDLINE%** - 명령 프로세서를 호출하는 원본 명령줄로 확장합니다

# 배치 파일

## .cmd 파일

윈도98에는 없지만, 윈도우2000이나 윈도우XP에는 **.cmd** 라는 [**확장자**](http://mwultong.blogspot.com/2006/04/blog-post_13.html)를 가진 텍스트 파일이 있습니다. 이 파일의 '파일 형식'은 **Windows NT 명령어 스크립트** 입니다.

그런데 이것은 도스의 배치 파일, 즉 **MS-DOS 일괄 파일** 과 사실상 똑같은 것입니다. 다만 도스에서는 호환되지 않는 문법이 사용될 수 있습니다. 가령 도스에서는, CD 명령으로, 다른 드라이브의 디렉토리로 직접 이동할 수 없습니다. 그러나 윈도우2000 이상의 운영체제 즉 윈도NT계열에서는, CD에 /D 옵션을 붙이면 다른 드라이브의 디렉토리(폴더)로 직접 이동할 수 있습니다.

NT용 배치 파일을, 윈도98등에서 실행하면 위험하기도 하고 잘 실행되지도 않습니다. 그래서 확장자를 cmd 로서 구분한 것으로 알고 있습니다.

.cmd 파일은 .bat 파일의 문법과 사실상 똑같습니다.

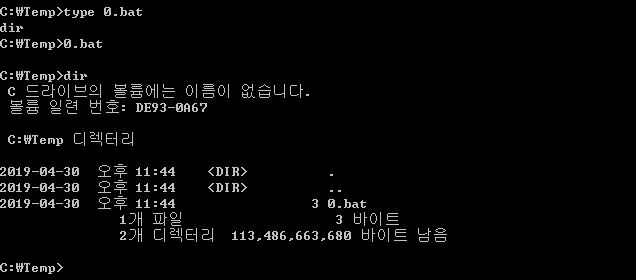
.cmd 파일은 윈도98이나 도스(MS-DOS 6.0 등)에서는 실행되지 않습니다. 만약 .bat 으로 확장자를 바꾸면 실행할 수 있지만, 그러면 위험합니다. 왜냐하면 .cmd 파일에는 윈도98이 인식할 수 없거나 오작동할 수 있는 새로운 문법과 명령어가 포함되어 있을 수 있기 때문입니다.

따라서 윈도98에서 실행되지 않도록 배치 파일을 작성하려면 cmd 확장자를 붙이면 됩니다.

## @echo off 와 @echo on

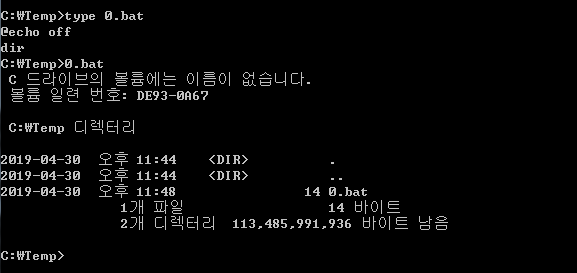
@echo off 는, 배치 파일이 명령들을 쓸데없이 '복창'하지 않도록 만들어서, 화면을 간결하게 하기 위한 명령입니다.

이렇게 dir 명령 하나만 든 간단한 배치 파일을 만들어 0.bat 등의 이름으로 저장하고 실행하면



위와 같이 dir 명령이 불필요하게 화면에 출력됩니다. 현재 실행 중인 명령의 이름까지 나올 필요는 없습니다. 결과만 나오면 됩니다.

그러나 @echo off 를 붙이면



이제 dir 이라는 글자는 나타나지 않고 dir의 결과만이 나오기에 화면이 간결해졌습니다.

배치 파일 안에는 수많은 명령어들이 있습니다. 배치 파일이 그 명령어들을 하나 실행시킬 때마다, 그 명령어들 자체도 화면에 출력한다면 산만해지게 됩니다. echo off 는 명령어의 결과만을 출력하라는 명령입니다.

그런데 echo off 자체도 하나의 명령이기 때문에, 이 echo off 명령 자체가 화면에 쓸데없이 출력됩니다. 배치 파일에서는, 어떤 명령어 앞에 골뱅이(@)를 하나 붙이면 그 명령어에 대해 '복창'하지 말라는 의미이기 때문에, **@echo off** 라고 하면 이제, 완벽하게 모든 복창을 금지시킬 수 있습니다.

@echo off 와 @echo on

배치 파일의 첫 줄에 항상 **@echo off** 를 추가하는 것이 좋습니다. 그런데 드문 경우지만, 명령어의 복창이 필요할 때가 있습니다. 그럴 때는 복창이 필요한 부분에 @echo on 을 간단히 추가하면 됩니다. (OFF가 아닌 ON 입니다.) 복창의 필요성이 없는 부분에서는 다시 @echo off 를 추가하면 됩니다.

즉 @echo off 와 @echo on 은, 배치 파일의 꼭 첫줄에 있지 않아도 되고, 필요한 부분에 마음껏 추가하여 에코를 켜고 끌 수 있습니다.

## CALL 명령

배치파일 안에서 다른 배치파일을 호출하여 실행하려면 **CALL** 명령을 쓰면 됩니다. 그러면 그 외부 배치파일의 전부가 실행됩니다. 그런데 다른 배치파일의 특정 라벨로 뛰어넘으려면, 즉 외부 배치파일의 일부만 실행시키려면 **CALL** 과 **GOTO** 명령을 동시에 사용해야 합니다.

현재 배치 파일에

call "다른 배치파일명" "라벨명"

이렇게 하고, 호출될 다른 배치 파일의 첫 부분에

goto %1

이렇게 쓰면 됩니다.

다음 예제를 보면 쉽게 알 수 있습니다.

A.bat 안에서 B.bat의 :GOOGLE 이라는 라벨(레이블)도 직접 이동한 후,

다시 A.bat 으로 돌아오는 것입니다.

파일명: A.bat

|  |
| --- |
| @echo off  call B.bat GOOGLE  echo.  echo ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::  echo B.bat 의 GOOGLE 이라는 라벨로 이동한 후,  echo 다시 A.bat으로 돌아와서 끝납니다.  echo :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: |

파일명: B.bat

|  |
| --- |
| @echo off  goto %1  :YAHOO  echo 여기는 '야후' 라벨입니다  goto QUIT  :GOOGLE  echo 여기는 '구글' 라벨입니다  goto QUIT  :MSN  echo 여기는 '엠스엔' 라벨입니다  goto QUIT  :QUIT  echo 여기서 A.bat으로 다시 돌아갑니다. |

실행 결과:

|  |
| --- |
| D:\Z>A.bat  여기는 '구글' 라벨입니다  여기서 A.bat으로 다시 돌아갑니다.  ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::  B.bat 의 GOOGLE 이라는 라벨로 이동한 후,  다시 A.bat으로 돌아와서 끝납니다.  ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::  D:\Z> |

## 주석 처리

배치파일에서는 rem 이라는 예약어로 주석 처리를 할 수 있습니다. 어떤 코드나 문자열 앞에 rem 을 붙이면 주석문(remarks)이 되어 설명을 마음대로 적을 수 있습니다. 즉, rem 이라는 문자열 뒤쪽은 실행에서 무시됩니다.

**배치파일(일괄파일) 주석 처리 방법, 주석 붙이기**

|  |
| --- |
| @echo off  ​  rem 이것은 배치 파일에 설명문을 rem 붙이는 간단한 예제입니다.  dir  : 이것은 배치 파일에 설명문을  : 붙이는 간단한 예제입니다.  dir  :: 이것은 배치 파일에 설명문을  :: 붙이는 간단한 예제입니다.  dir |

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

:: 이것은 배치 파일에 설명문을

:: 붙이는 간단한 예제입니다.

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

dir

**변칙적인 주석문**

배치파일의 GOTO문에서 라벨로 사용되는 콜론(:)을 주석 처리 기호로 사용할 수도 있습니다. 위의 예제의 2, 3, 4번째 경우입니다. 이러면 보기도 깔끔하고 타이핑도 쉽지만 IF문 속에서 사용한다거나 할 때는 에러가 나더군요.

3번째 경우는 콜론 2개를 연속으로 :: 이렇게 쓰는 것입니다.

배치 파일을 문법 강조\*할 때, 콜론 1개는 주석으로 인정하지 않습니다. 사실 주석이 아니기 때문입니다. 그런데 :: 이렇게 2개를 쓰면 주석으로 인정해 줍니다.

\* 울트라에디터에서 배치 파일을 문법 강조하려면 울트라에디트 사이트에 가서 배치 파일 문법 강조용 파일을 다운로드받은 후, 울트라에디터 설치 폴더 안에 있는 wordfile.txt 파일을 편집하여 넣어주어야 합니다. 그래서 좀 복잡합니다. 다만, [**에디터**](http://mwultong.blogspot.com/2006/06/vim-vi-gvim.html)는 배치 파일도 기본적으로 문법 강조를 해 줍니다.

4번째 경우처럼

:::::::::::::::::::::::::::::::

콜론을 여러 개 붙여서 선을 그어 주어도 됩니다.

**결론**

**rem** 을 앞에 붙이거나, 또는 **::** 을 앞에 붙이면, 그 행이 모두 주석이 됩니다

# 참고 사이트

* <https://shinb.tistory.com/405> [신비 블로그]