

부록
Linux 기본 명령어
&
Hadoop, R 및 R-studio 설치

최 우 석 & 김 석 찬

12143543@inha.edu 12160538@inha.edu

부록A. Linux 기본 명령어

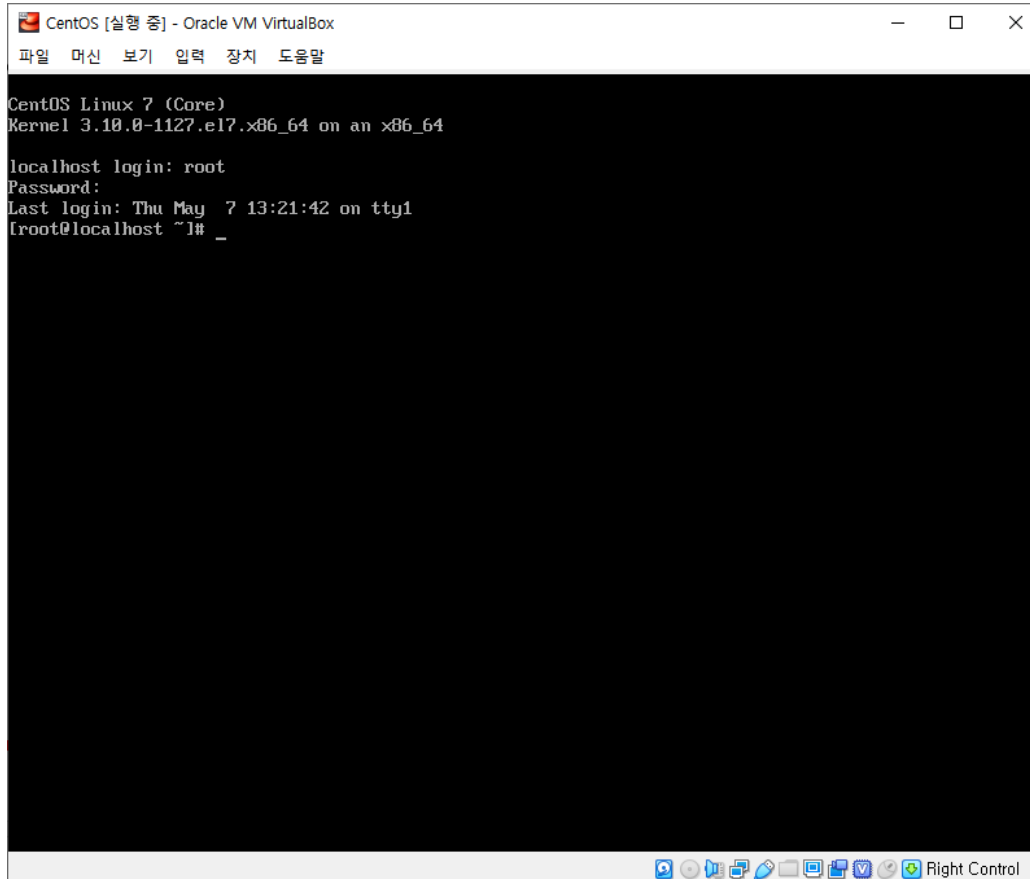
- ✓ Linux OS를 원활히 사용하기 위한 주요 명령어들을 정리하였음
- ✓ 15개의 명령어와 더불어 Linux 내 기타 기능들까지 추가로 정리하였음
- ✓ 대부분의 설명에선 필자가 CUI환경에서 실습을 진행하였음(여러 설명 사진 속 모습이 학습자의 환경과 달라 보일 수 있음. 겉으로만 달라 보이는 것 뿐이기 때문에, 사진 속 모습을 그대로 따라하여도 무방함)
- ✓ GUI환경(Window환경)에서 명령어를 입력할 수 있는 창(터미널)을 실행시키려면, '프로그램 - 시스템 도구 - 터미널'을 클릭하면 됨

A-1. ls

- ✓ ls는 현재 디렉토리(폴더) 내 파일의 목록을 보여주는 명령어임
- ✓ 해당 명령어와 함께 사용되는 네 개의 옵션들에 대해서도 추가적으로 소개하겠음

A-1. ls

- ✓ Linux를 실행시키고 로그인을 하면, 다음과 같은 화면이 나타남

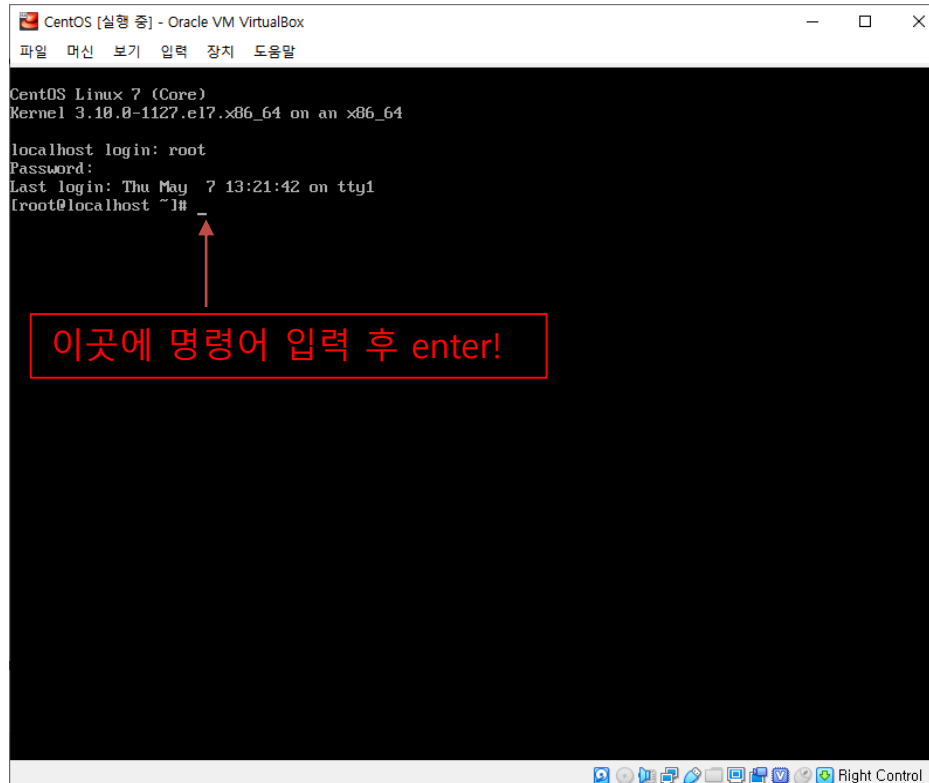


```
CentOS Linux 7 (Core)
Kernel 3.10.0-1127.el7.x86_64 on an x86_64

localhost login: root
Password:
Last login: Thu May 7 13:21:42 on tty1
[root@localhost ~]# _
```

A-1. ls

- ✓ '#' 또는 '&'기호 옆 키보드 커서에 'ls'라는 명령어를 입력하고 enter를 누르면 해당 명령어가 실행됨(root 계정으로 login한 경우 '#'기호가 나타나고, root 계정 외 다른 계정으로 login한 경우 '\$'기호가 나타남)



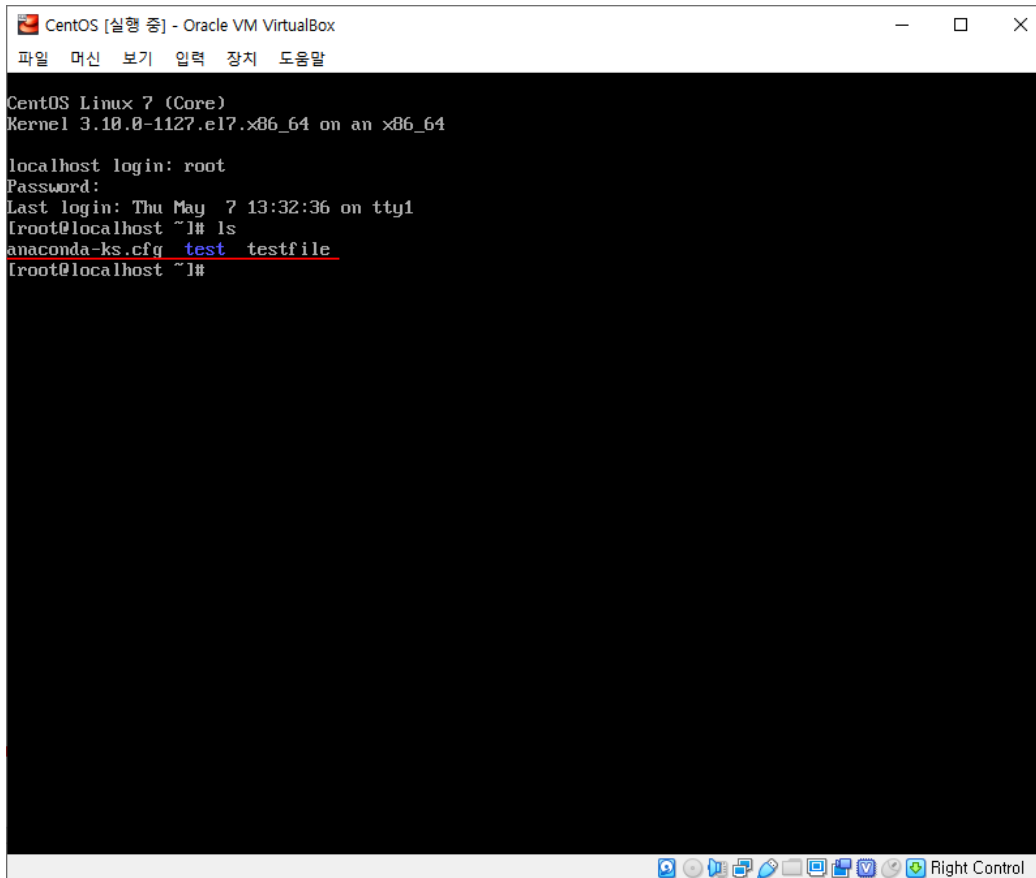
```
CentOS Linux 7 (Core)
Kernel 3.10.0-1127.el7.x86_64 on an x86_64

localhost login: root
Password:
Last login: Thu May 7 13:21:42 on tty1
[root@localhost ~]#
```

이곳에 명령어 입력 후 enter!

A-1. ls

- ✓ 해당 명령어를 실행하면, 현재 사용자가 위치한 디렉토리 내 파일들의 목록이 나타남
- ✓ 밑의 'test'처럼 파란색으로 나타나는 것은 디렉토리임



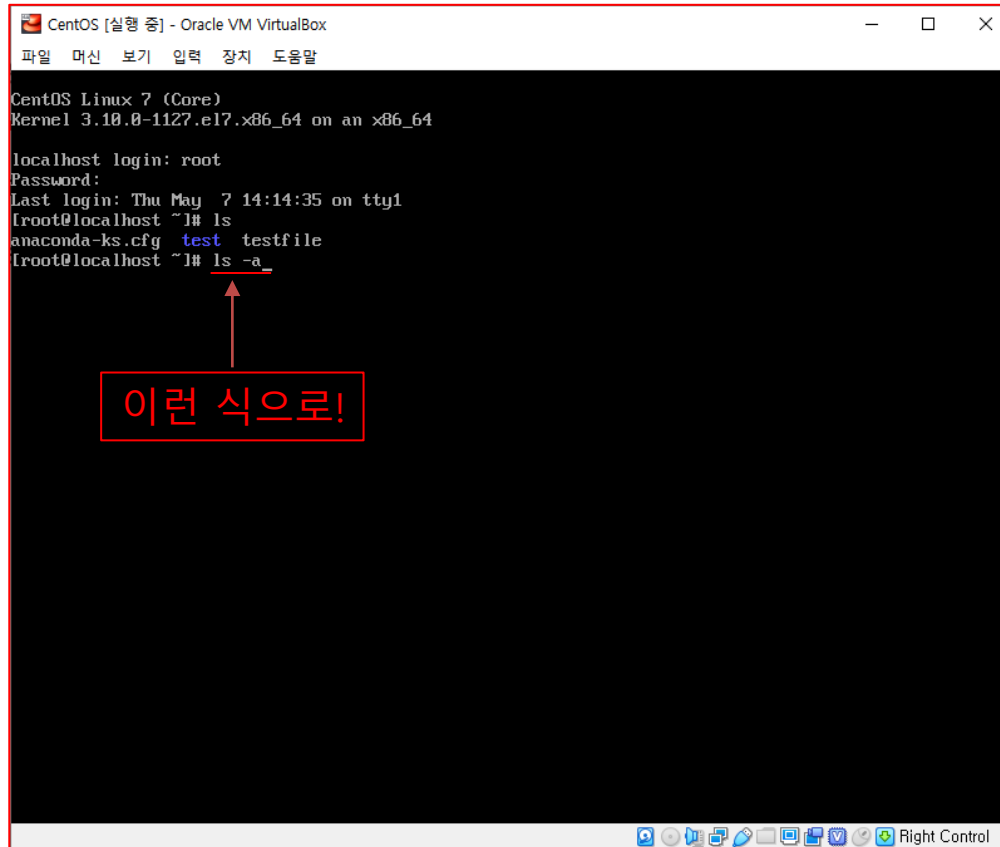
```
CentOS [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

CentOS Linux 7 (Core)
Kernel 3.10.0-1127.el7.x86_64 on an x86_64

localhost login: root
Password:
Last login: Thu May  7 13:32:36 on tty1
[root@localhost ~]# ls
anaconda-ks.cfg  test  testfile
[root@localhost ~]#
```

A-1. ls

- ✓ 특정 명령어 뒤에 '-옵션'을 입력하면, 해당 명령어에 옵션이 적용되어 실행됨
(ex : ls -a)



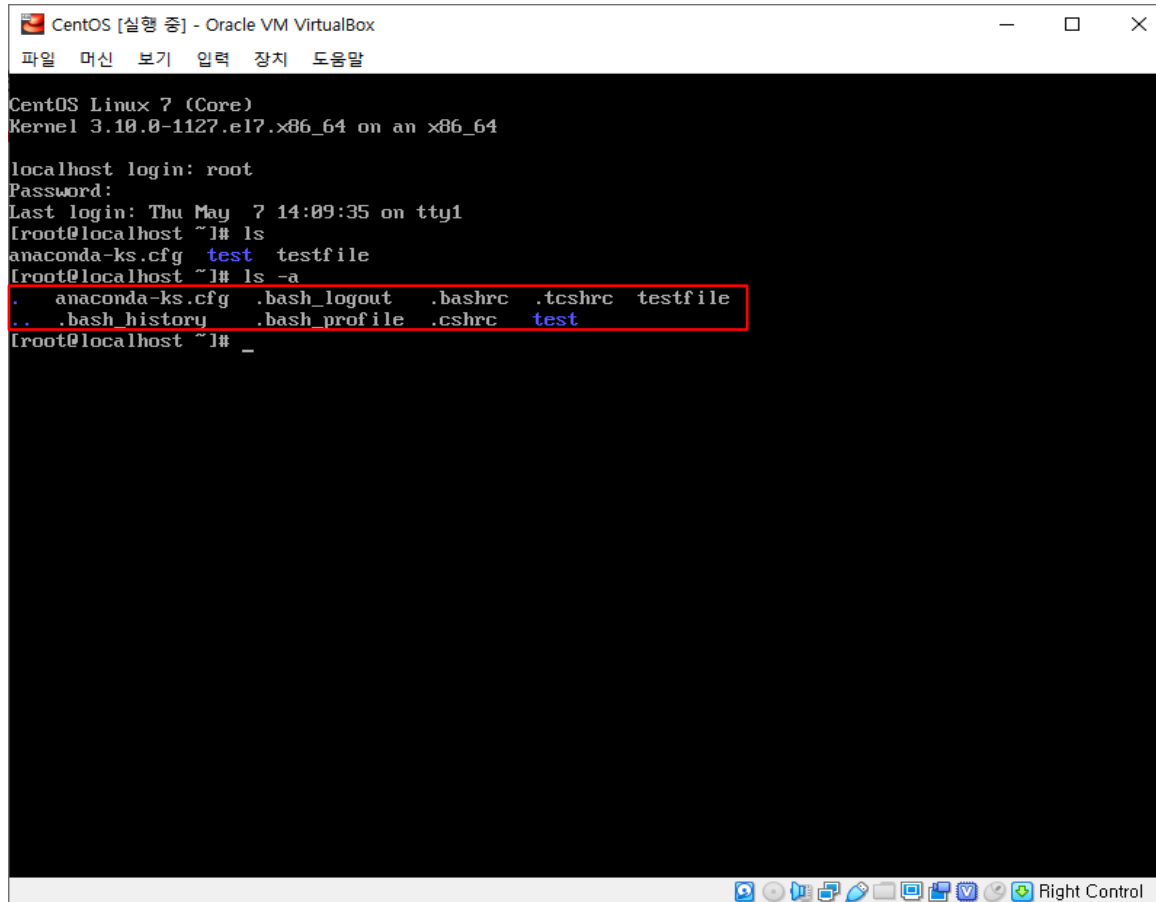
```
CentOS Linux 7 (Core)
Kernel 3.10.0-1127.el7.x86_64 on an x86_64

localhost login: root
Password:
Last login: Thu May 7 14:14:35 on tty1
root@localhost ~]# ls
anaconda-ks.cfg  test  testfile
root@localhost ~]# ls -a
```

이런 식으로!

A-1. ls

- ✓ -a 옵션은 디렉토리 내 dot파일을 포함한 모든 파일들의 목록을 나타냄



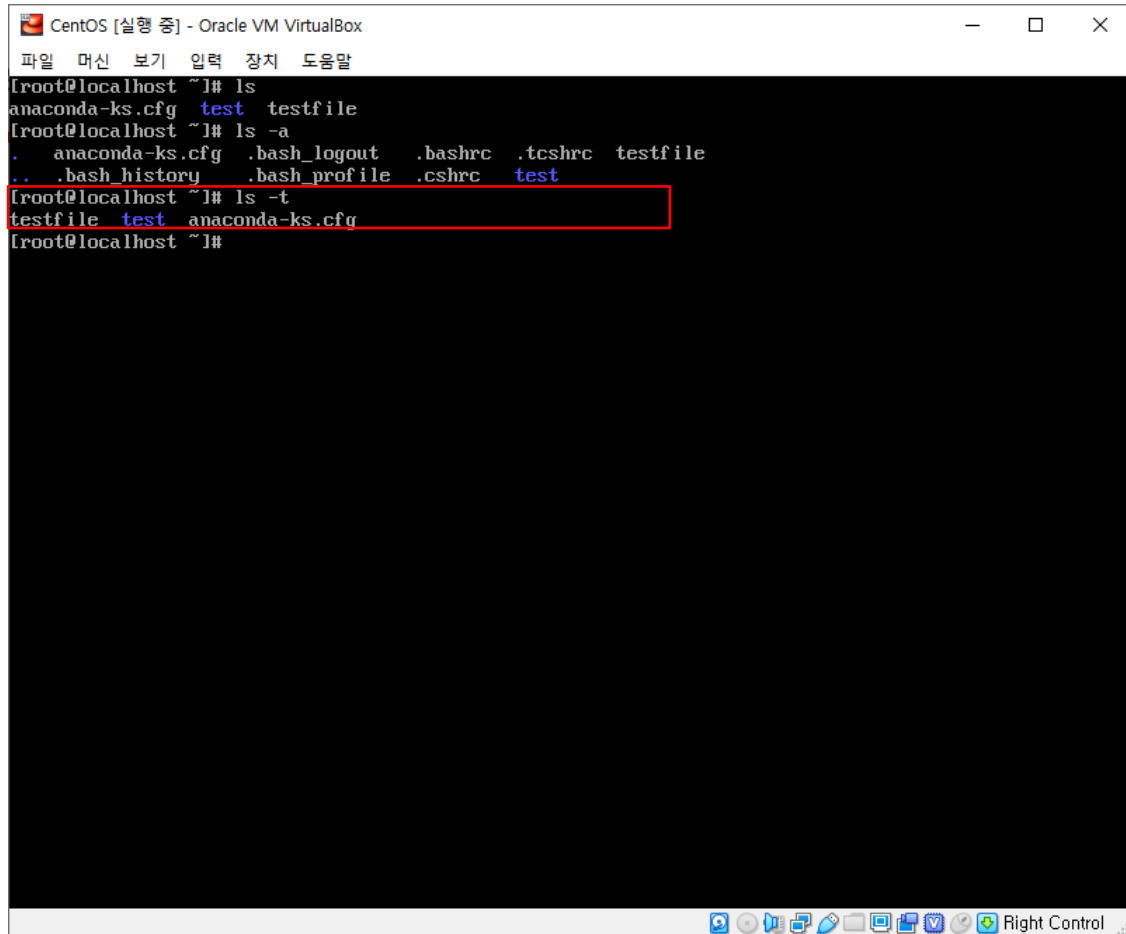
```
CentOS [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

CentOS Linux 7 (Core)
Kernel 3.10.0-1127.el7.x86_64 on an x86_64

localhost login: root
Password:
Last login: Thu May 7 14:09:35 on tty1
[root@localhost ~]# ls
anaconda-ks.cfg  test  testfile
[root@localhost ~]# ls -a
.  anaconda-ks.cfg  .bash_logout  .bashrc  .tcshrc  testfile
.. .bash_history  .bash_profile .cshrc   test
[root@localhost ~]# _
```


A-1. ls

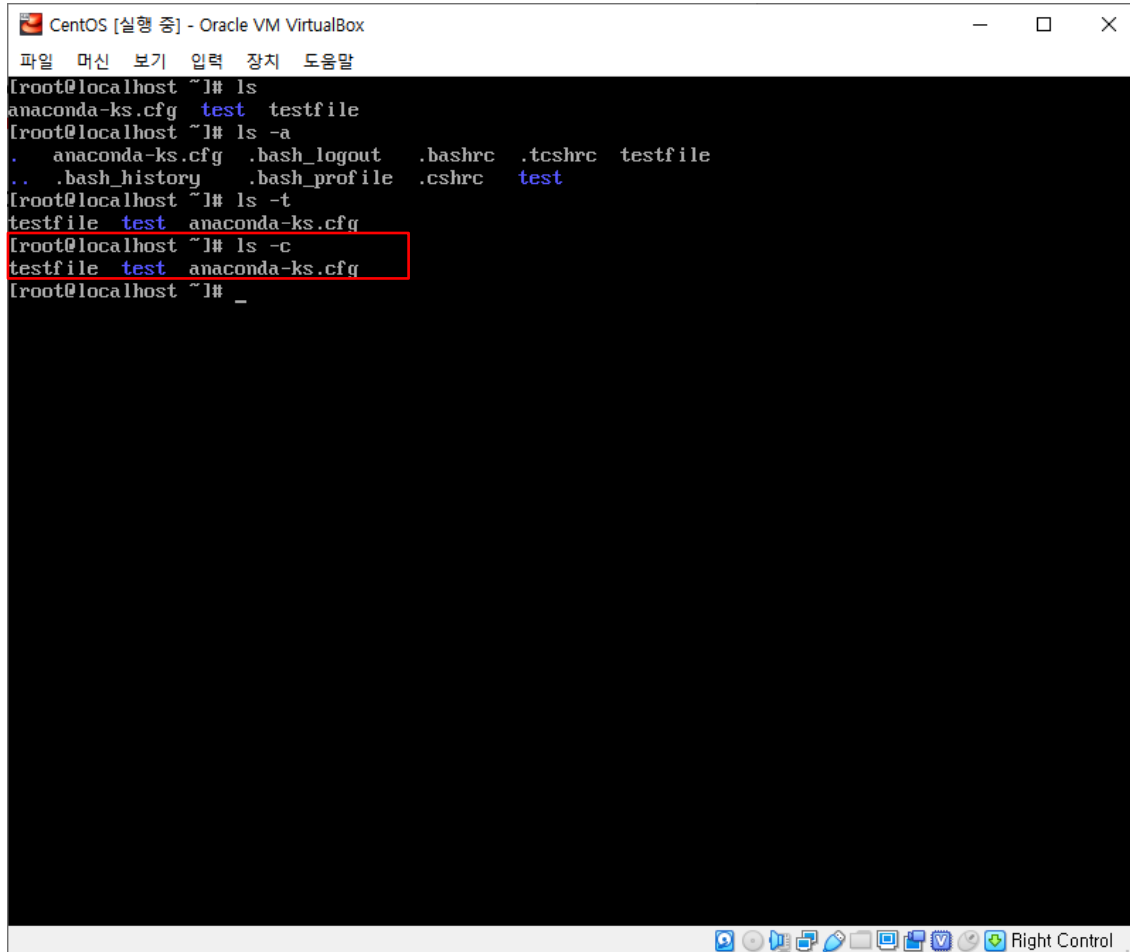
✓ -t 옵션은 파일이 생성된 순으로, 디렉토리 내 파일들의 목록을 나타냄



```
CentOS [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말
[root@localhost ~]# ls
anaconda-ks.cfg  test  testfile
[root@localhost ~]# ls -a
.  anaconda-ks.cfg  .bash_logout  .bashrc  .tcshrc  testfile
.. .bash_history    .bash_profile .cshrc    test
[root@localhost ~]# ls -t
testfile test anaconda-ks.cfg
[root@localhost ~]#
```

A-1. ls

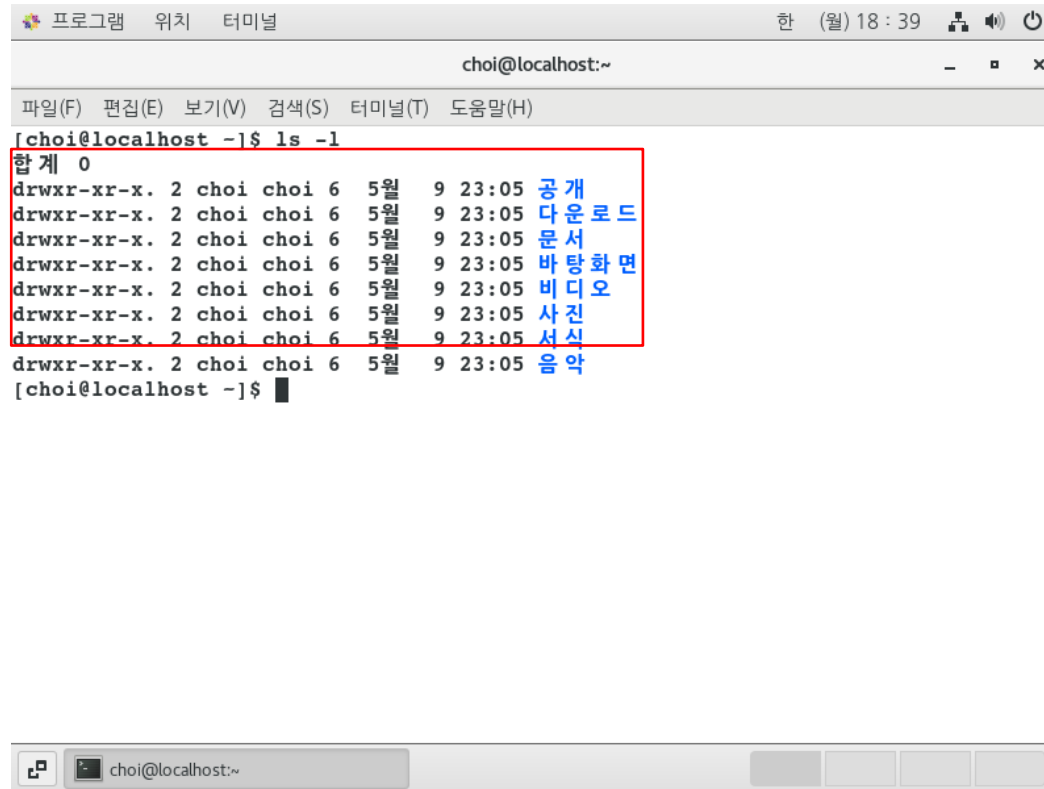
✓ -c 옵션은 파일이 수정된 순으로, 디렉토리 내 파일들의 목록을 나타냄



```
CentOS [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말
[root@localhost ~]# ls
anaconda-ks.cfg  test  testfile
[root@localhost ~]# ls -a
.  anaconda-ks.cfg  .bash_logout  .bashrc  .tcshrc  testfile
.. .bash_history  .bash_profile .cshrc   test
[root@localhost ~]# ls -t
testfile  test  anaconda-ks.cfg
[root@localhost ~]# ls -c
testfile  test  anaconda-ks.cfg
[root@localhost ~]# _
```

A-1. ls

✓ -l 옵션은 파일과 디렉토리에 대한 권한을 함께 나타냄



```

[choi@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 2 choi choi 6 5월 9 23:05 공개
drwxr-xr-x. 2 choi choi 6 5월 9 23:05 다운로드
drwxr-xr-x. 2 choi choi 6 5월 9 23:05 문서
drwxr-xr-x. 2 choi choi 6 5월 9 23:05 바탕화면
drwxr-xr-x. 2 choi choi 6 5월 9 23:05 비디오
drwxr-xr-x. 2 choi choi 6 5월 9 23:05 사진
drwxr-xr-x. 2 choi choi 6 5월 9 23:05 서식
drwxr-xr-x. 2 choi choi 6 5월 9 23:05 음악
[choi@localhost ~]$
  
```

❖ 출력 결과는 왼쪽부터 **파일종류 및 권한(퍼미션)**, 링크수, **사용자(소유자)**, **그룹**, 파일크기, 수정시간, **파일이름**을 나타냄

A-1. ls

- ✓ 특히, 출력 결과 중 '파일종류 및 권한'을 해석하는 법에 주목해야 함
- ✓ 밑의 사진과 같이, 해당 문자열을 네 부분으로 끊어서 해석해야 함

drwxr-xr-x.
① ② ③ ④

- ① '파일(-)'인지 '디렉토리(d)'인지를 구분하는 부분
 - ② 사용자(**u**ser)의 권한을 나타내는 부분
 - ③ 그룹(**g**roup)의 권한을 나타내는 부분
 - ④ 다른 사용자(**o**thers)의 권한을 나타내는 부분
- ✓ rwx는 각각 읽기(**r**ead), 쓰기(**w**rite), 실행(**e**xecute)를 의미함

A-1. ls

drwxr-xr-x.

① ② ③ ④

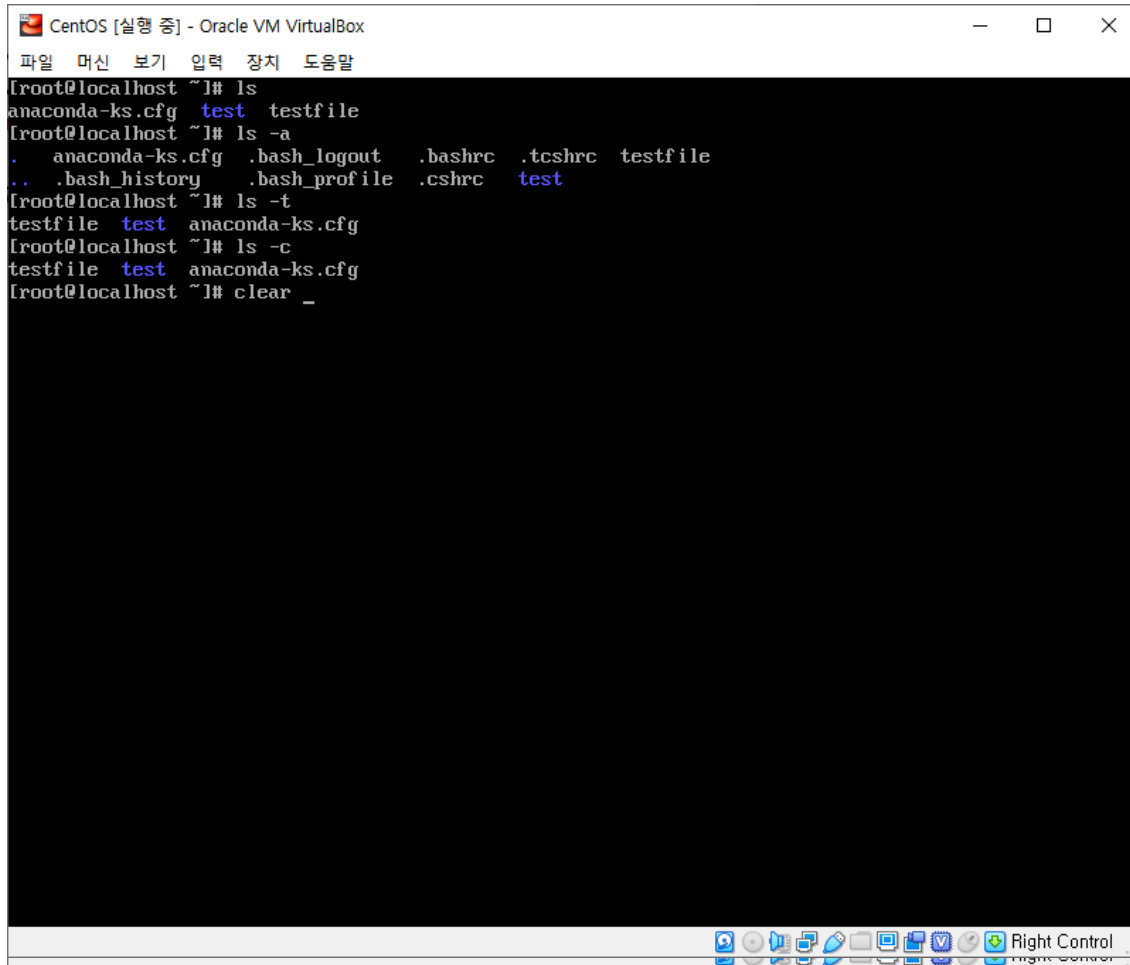
✓ 앞선 이론을 바탕으로 해당 예시를 해석해보면 다음과 같음

- ① 'd'라고 나와 있기 때문에, 이것은 디렉토리임
- ② 'rwx'라고 나와있기 때문에 사용자는 읽기, 쓰기, 실행의 권한을 전부 가지고 있음
- ③ 'r-x'라고 나와있기 때문에 그룹은 읽기, 쓰기의 권한을 가지고 있음
- ④ 'r-x'라고 나와있기 때문에 다른 사용자는 읽기, 쓰기의 권한을 가지고 있음

✓ 해당 부분을 해석할 줄 알아야만 'chmod'명령어를 제대로 활용할 수 있음
('chmod' 명령어는 뒷부분에 설명되어 있음)

A-2. clear

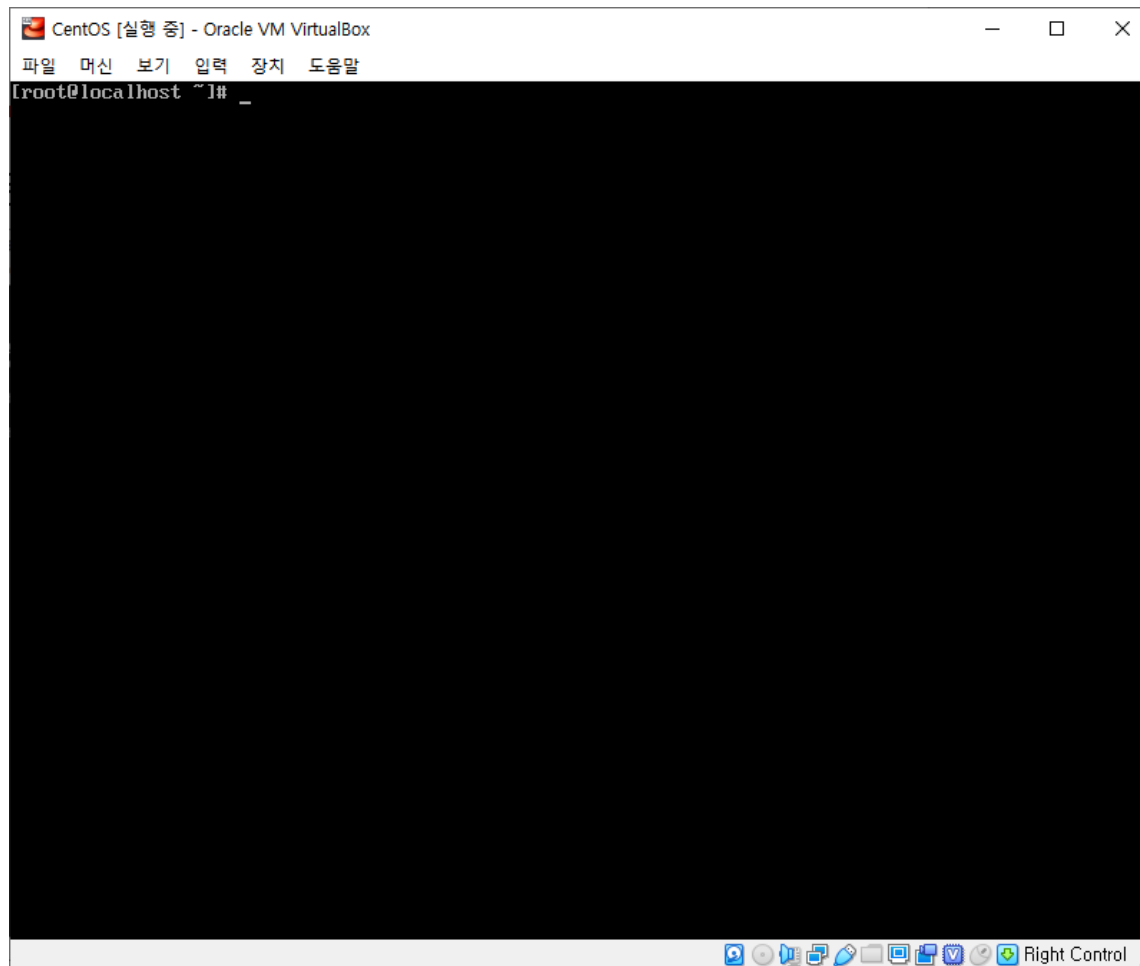
✓ Clear는 화면에 나타난 모든 내용을 지우는 명령어임



```
CentOS [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말
[root@localhost ~]# ls
anaconda-ks.cfg  test  testfile
[root@localhost ~]# ls -a
.  anaconda-ks.cfg  .bash_logout  .bashrc  .tcshrc  testfile
.. .bash_history  .bash_profile .cshrc   test
[root@localhost ~]# ls -t
testfile  test  anaconda-ks.cfg
[root@localhost ~]# ls -c
testfile  test  anaconda-ks.cfg
[root@localhost ~]# clear _
```

A-2. clear

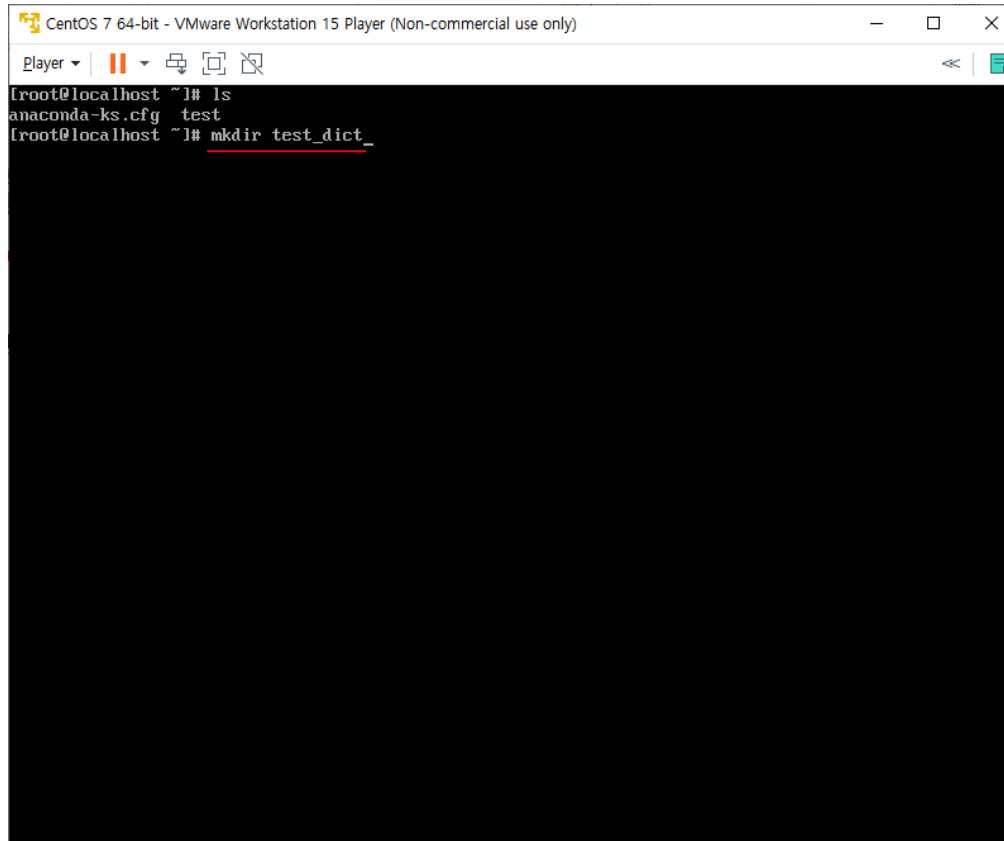
✓ Clear명령어 실행 결과



```
CentOS [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말
[root@localhost ~]# _
```

A-3. mkdir

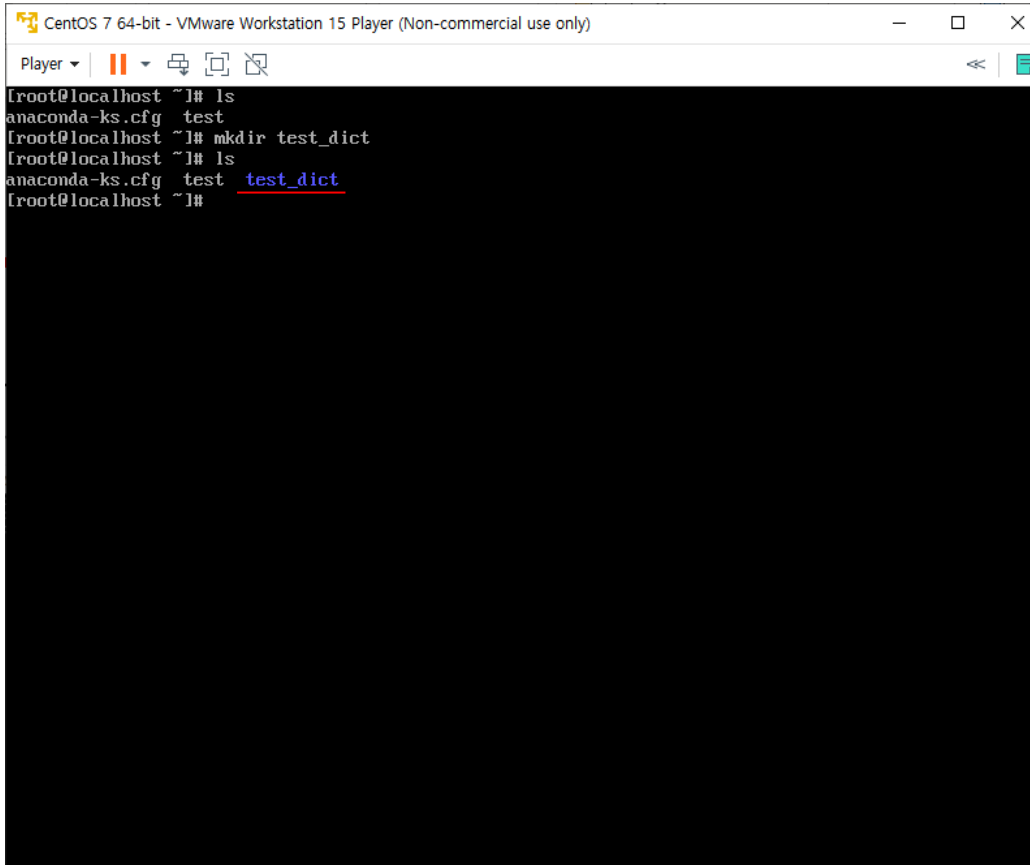
- ✓ mkdir은 디렉토리(폴더)를 생성하는 명령어임
- ✓ 'mkdir 폴더명'이런 형식으로 입력하면 됨



```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
[root@localhost ~]# ls
anaconda-ks.cfg  test
[root@localhost ~]# mkdir test_dict_
```


A-3. mkdir

- ✓ 해당 명령어를 입력한 후 ls 명령어로 현재 디렉토리 내 파일 목록을 확인하면, 방금 생성한 폴더를 확인할 수 있음(디렉토리는 파란색으로 표시됨)



```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
[root@localhost ~]# ls
anaconda-ks.cfg  test
[root@localhost ~]# mkdir test_dict
[root@localhost ~]# ls
anaconda-ks.cfg  test  test_dict
[root@localhost ~]#
```

A-4. cd

- ✓ cd는 현재 위치한 디렉토리에서 다른 디렉토리로 이동할 때 사용되는 명령어임
- ✓ 즉, Window OS에서 마우스로 특정 폴더를 더블 클릭하여 해당 폴더에 들어가는 것을 cd라는 명령어로 실시하는 것임
- ✓ cd 명령어로 다른 디렉토리로 이동하는 방법은 두 가지가 있음

- 상대경로를 이용하여 이동하는 방법

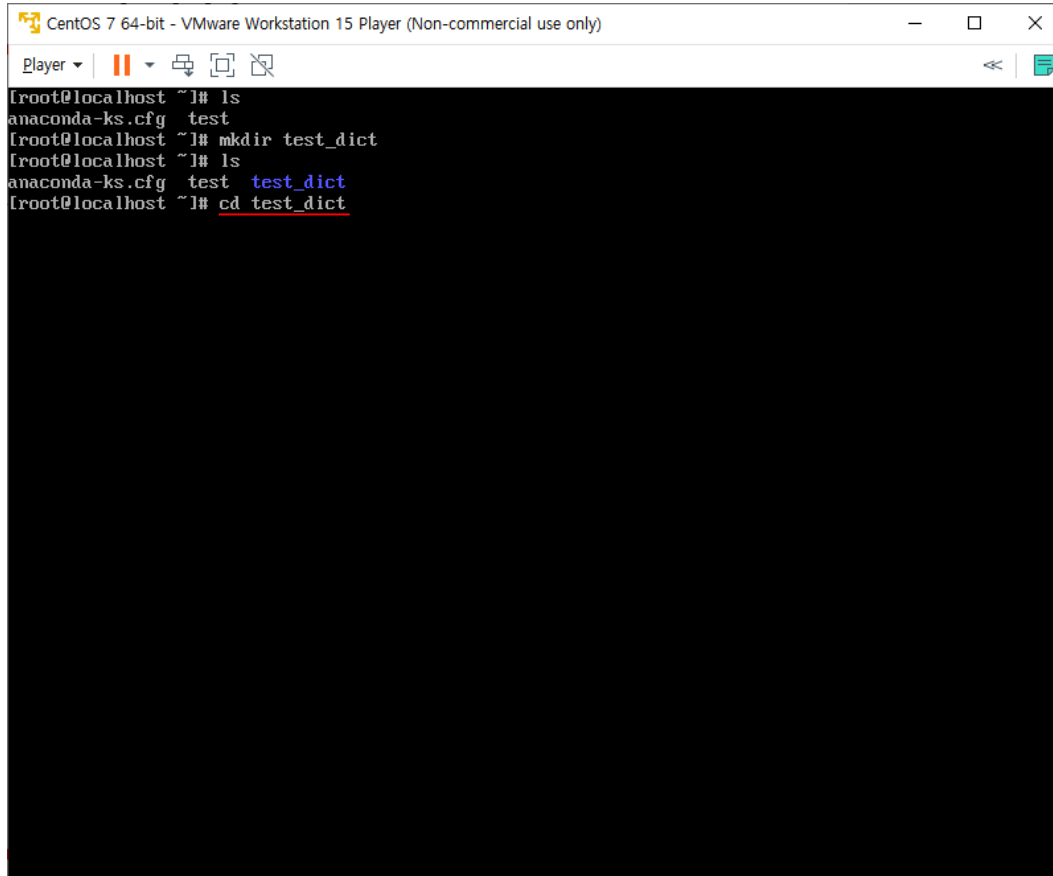
현재 사용자가 위치한 디렉토리를 기준으로 다른 디렉토리로 이동하는 방법

- 절대경로를 이용하여 이동하는 방법

"/"디렉토리(최상위 디렉토리)를 기준으로 다른 디렉토리로 이동하는 방법

A-4. cd

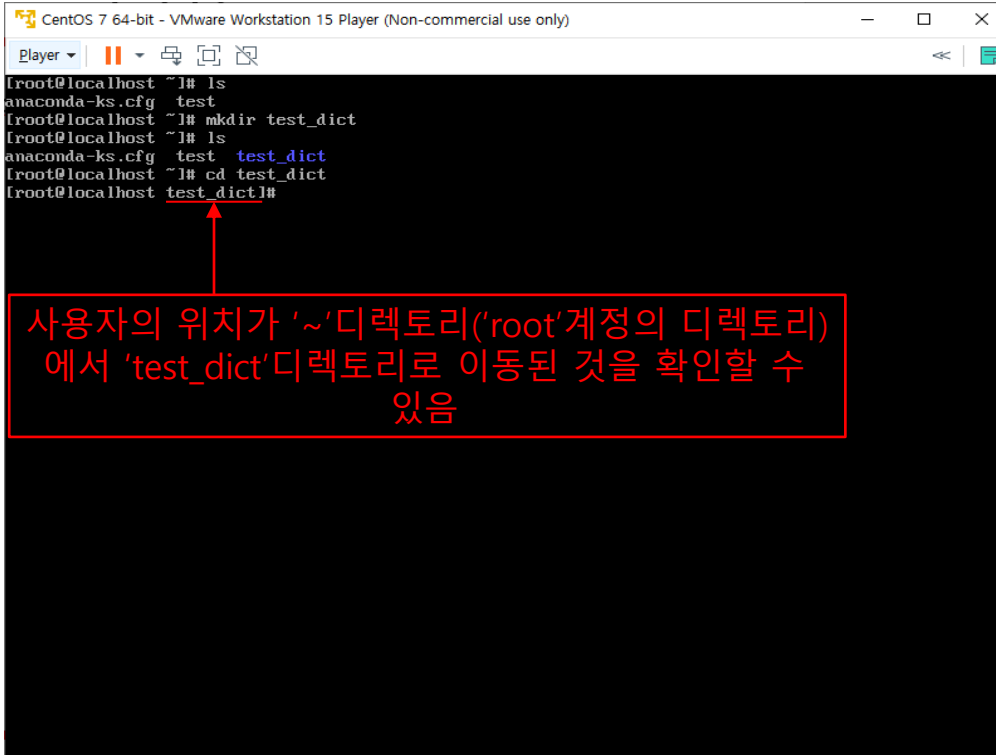
- ✓ 상대경로를 이용하여 특정 디렉토리로 이동하는 방법은 현재 사용자가 위치한 디렉토리를 기준으로 'cd 디렉토리경로'와 같은 형식으로 입력하면 됨



```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player | [Pause] [Full Screen] [Reset] [Undo] [Redo]
[root@localhost ~]# ls
anaconda-ks.cfg  test
[root@localhost ~]# mkdir test_dict
[root@localhost ~]# ls
anaconda-ks.cfg  test  test_dict
[root@localhost ~]# cd test_dict
```

A-4. cd

- ✓ 상대경로를 이용하여 'test_dict' 디렉토리로 이동한 결과



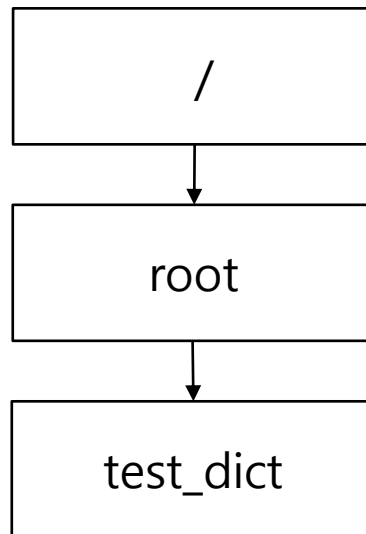
```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
[root@localhost ~]# ls
anaconda-ks.cfg  test
[root@localhost ~]# mkdir test_dict
[root@localhost ~]# ls
anaconda-ks.cfg  test  test_dict
[root@localhost ~]# cd test_dict
[root@localhost test_dict]#
```

사용자의 위치가 '~' 디렉토리('root' 계정의 디렉토리)에서 'test_dict' 디렉토리로 이동된 것을 확인할 수 있음

- ❖ 참고로 Linux OS에선 특정 계정을 생성하면, 해당 계정에 할당되는 디렉토리가 자동적으로 생성됨
- ❖ 예를 들어, 'choi'라는 계정을 생성하면 'choi'라는 디렉토리가 자동적으로 생성됨
- ❖ 특정 계정으로 로그인한 후 해당 계정에 할당된 디렉토리로 이동하면, 이 디렉토리는 '~' 디렉토리로 표기됨

A-4. cd

- ✓ 절대경로를 이용하여 특정 디렉토리로 이동하는 방법은 'cd /디렉토리명/디렉토리명/디렉토리명/.....' 과 같은 형식으로 입력하면 됨
- ✓ 절대경로를 이용할 때 핵심은 반드시 '/'디렉토리부터 차례대로 경로입력을 해줘야 한다는 것임
- ✓ 'test_dict'디렉토리로 예를 들자면, 해당 디렉토리의 구체적인 경로는 다음과 같음



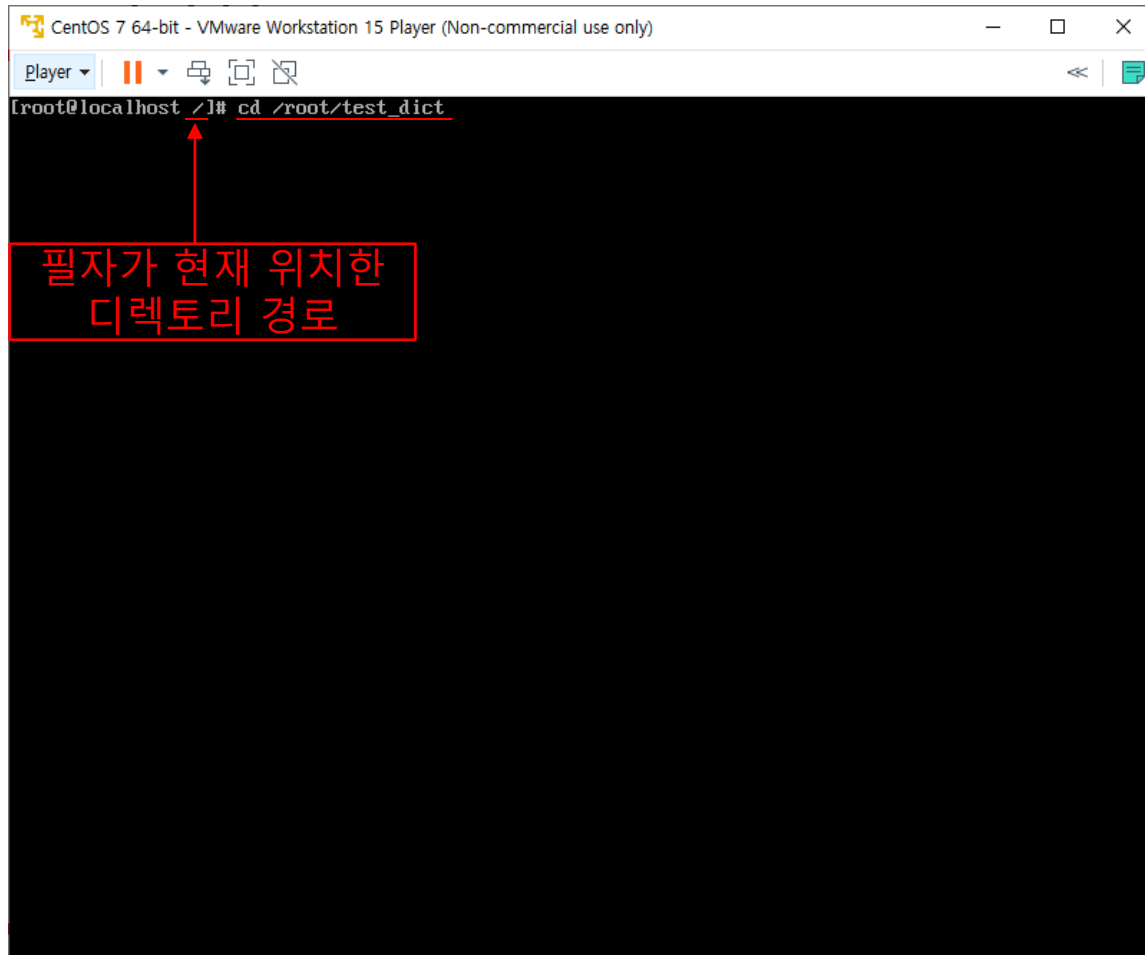
- ✓ 즉, '/'디렉토리 하위에 'root'디렉토리가 존재하고, 'root'디렉토리 하위에 'test_dict'디렉토리가 존재하는 것임

A-4. cd

- ✓ 결국, 'cd /root/test_dict'라는 명령어를 입력하면 절대경로를 이용하여 'test_dict' 디렉토리로 이동할 수 있음
- ✓ 경로를 기입할 때 '~하위의'라는 표현을 '/' 기호로 나타냄
- ✓ 예를 들어, 'root 디렉토리 하위의 test_dict 디렉토리'를 기입할 때는 'root/test_dict'라고 기입함

A-4. cd

✓ 'cd /root/test_dict'라고 입력

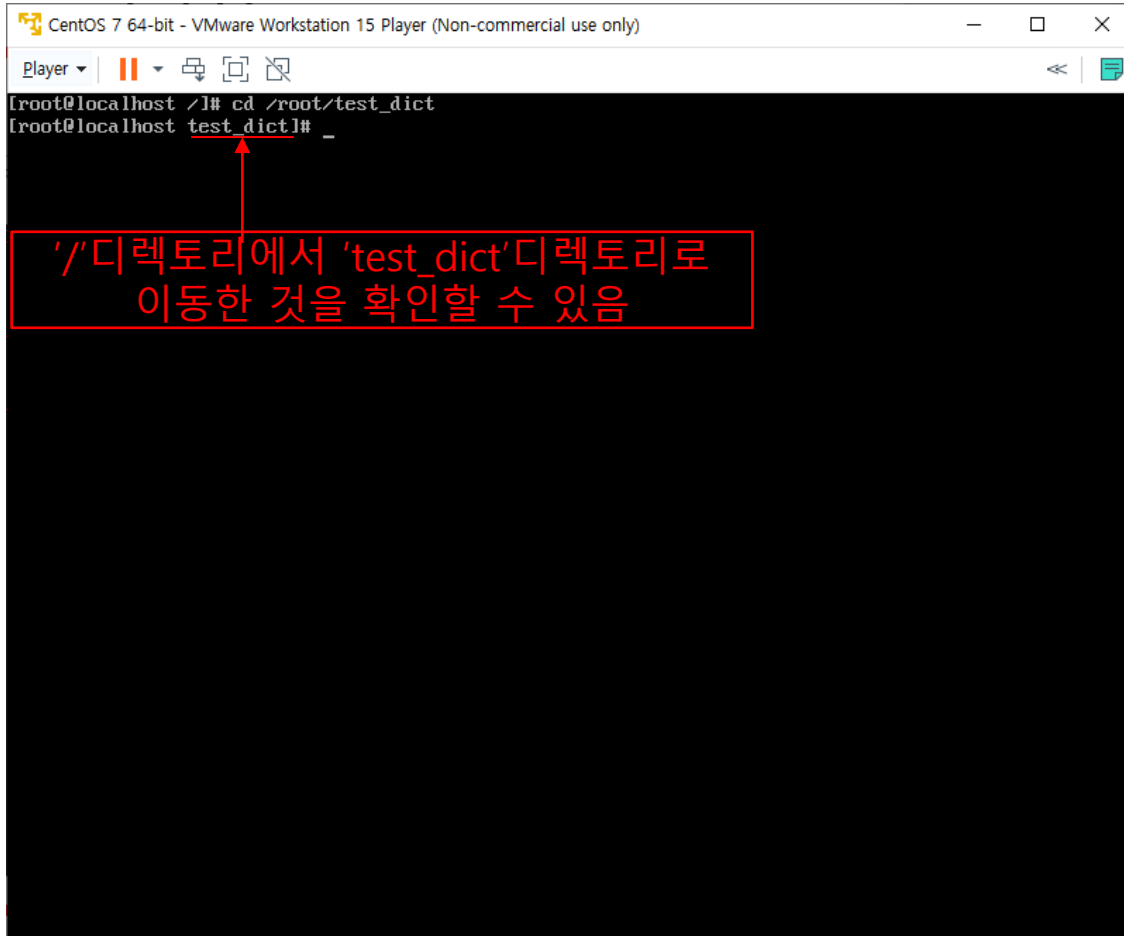


```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
[root@localhost ~]# cd /root/test_dict
```

필자가 현재 위치한
디렉토리 경로

A-4. cd

✓ 해당 명령어 실행 결과



```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player | || | |
[root@localhost /]# cd /root/test_dict
[root@localhost test_dict]# _
```

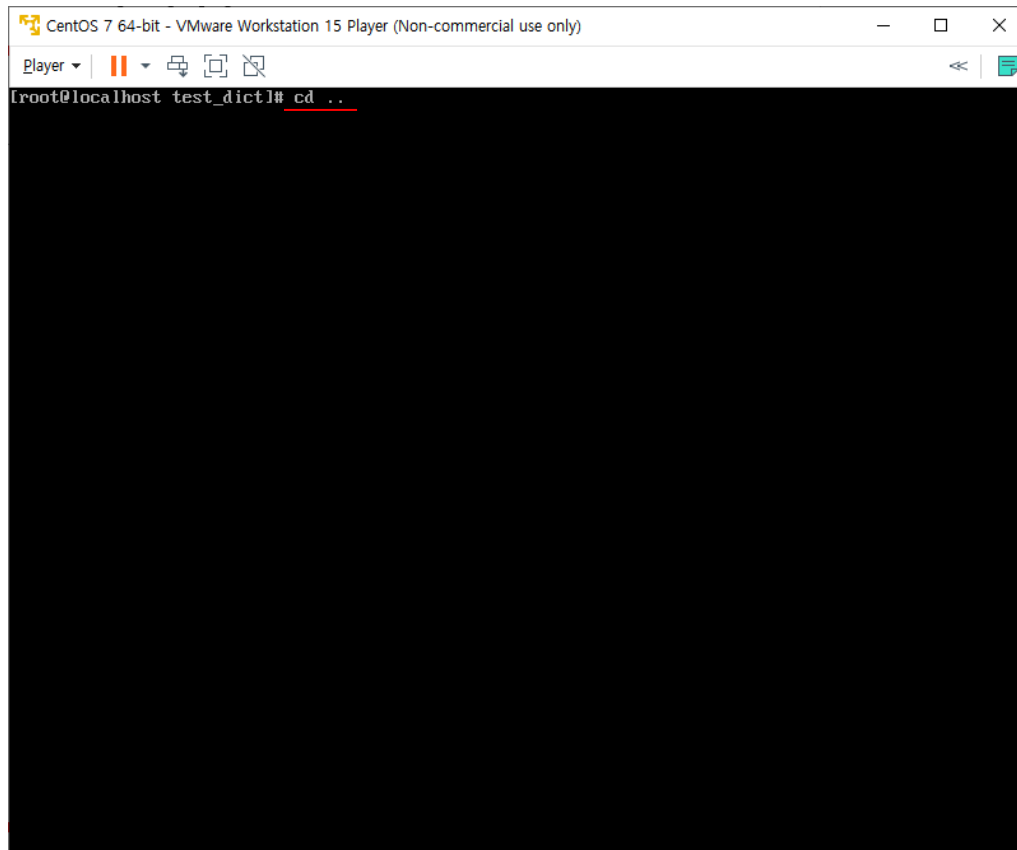
'/'디렉토리에서 'test_dict'디렉토리로
이동한 것을 확인할 수 있음

A-4. cd

- ✓ 'cd 이전 디렉토리명'을 통해 사용자가 현재 위치한 디렉토리에서 이전 디렉토리로 이동할 수 있음
- ✓ 하지만 위의 방법 보다는 'cd ..'라는 명령어를 통해 이전 디렉토리로 더 쉽게 이동할 수 있음

A-4. cd

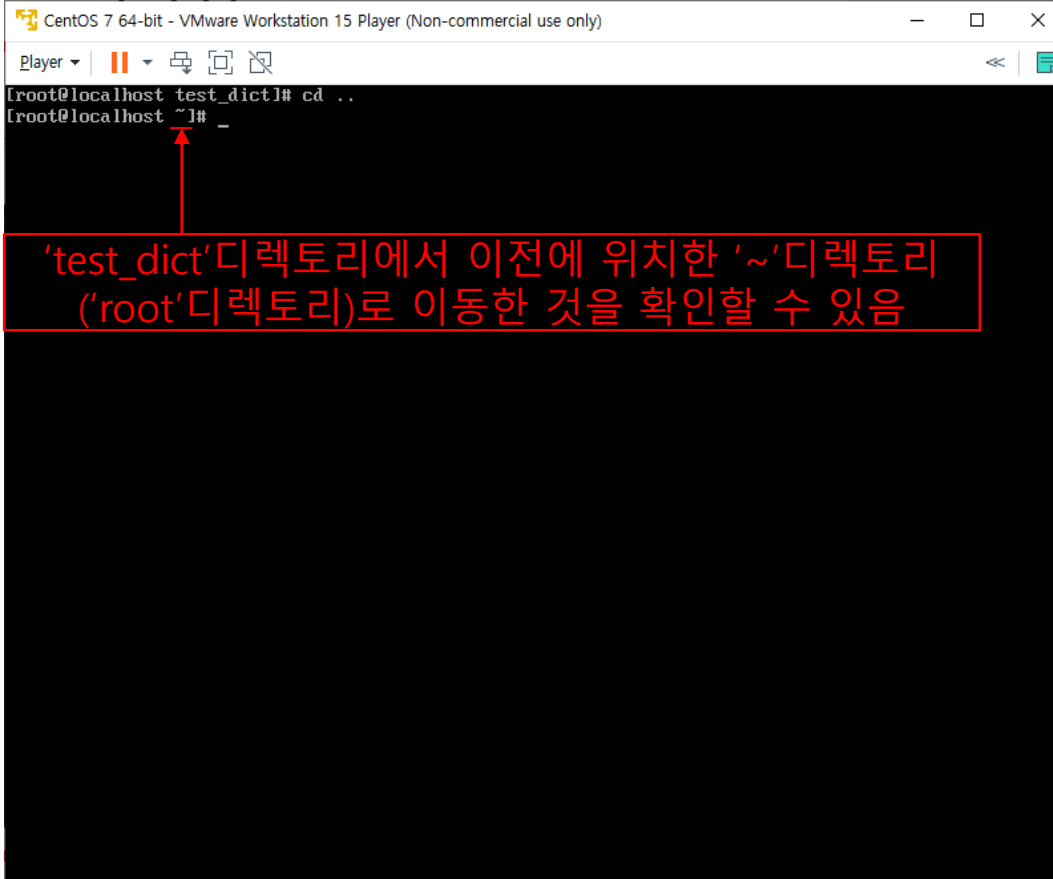
✓ 'cd ..' 명령어 입력



```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
[root@localhost test_dict]# cd ..
```

A-4. cd

✓ 해당 명령어 실행 결과



```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player | [Icons]
[root@localhost test_dict]# cd ..
[root@localhost ~]#
```

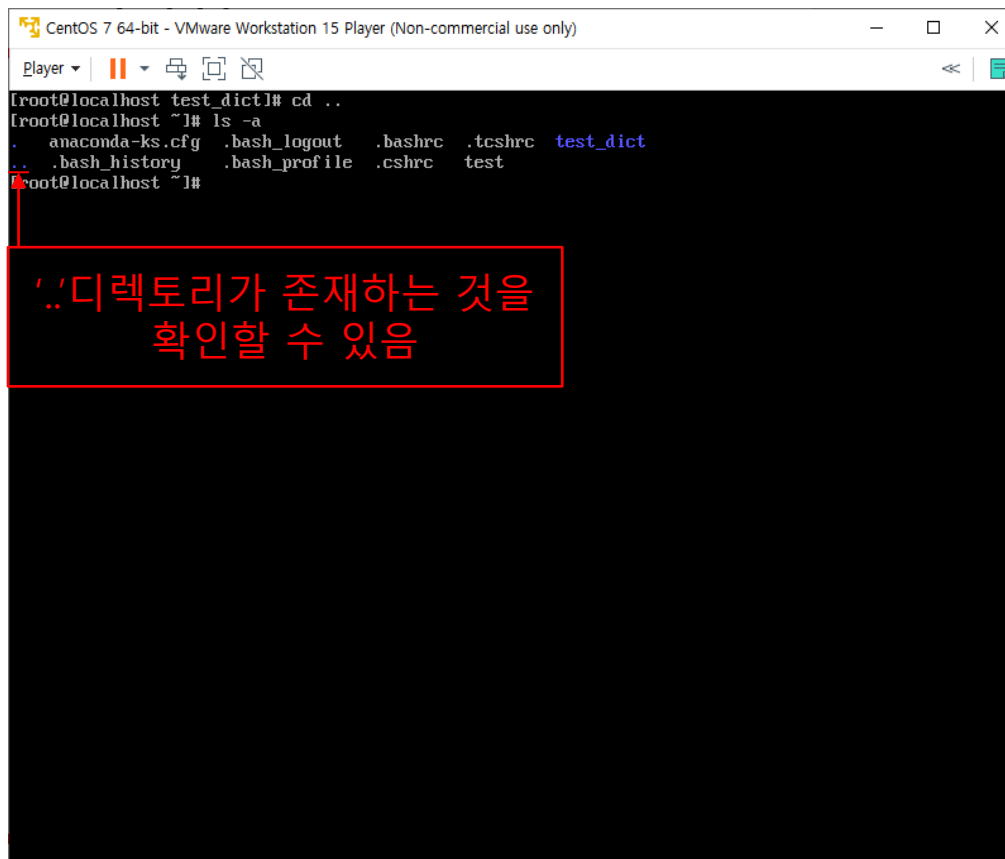
'test_dict'디렉토리에서 이전에 위치한 '~'디렉토리 ('root'디렉토리)로 이동한 것을 확인할 수 있음

A-4. cd

- ✓ 'cd ..'을 통해 이전 디렉토리로 이동할 수 있는 이유는 '/'디렉토리를 제외한 모든 디렉토리에 '..'라는 디렉토리가 존재하고, 이 디렉토리가 이전 디렉토리를 의미하고 있기 때문임
- ✓ 이를 앞서 설명했던 'ls -a'명령어를 통해 확인할 수 있음

A-4. cd

✓ 'ls -a' 명령어 실행 결과



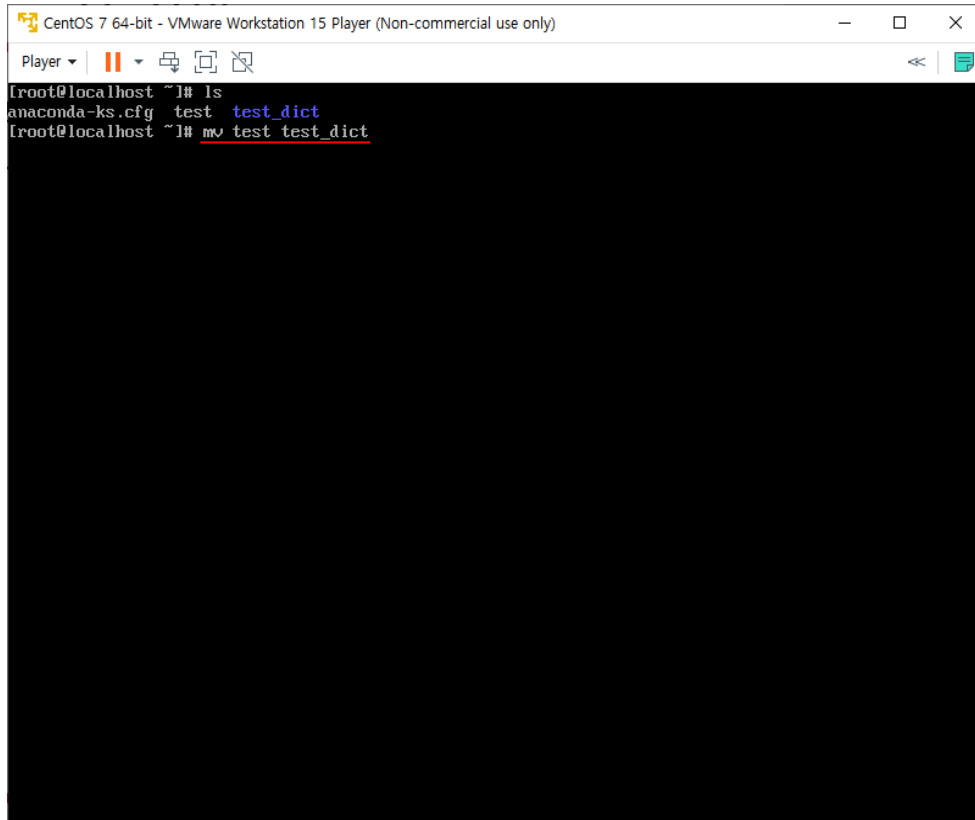
```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
[root@localhost test_dict]# cd ..
[root@localhost ~]# ls -a
.  anaconda-ks.cfg  .bash_logout  .bashrc  .tcshrc  test_dict
.. .bash_history    .bash_profile .cshrc   test
[root@localhost ~]#
```

'..'디렉토리가 존재하는 것을
확인할 수 있음

❖ 참고로 '..'디렉토리는 현재 디렉토리를 의미함

A-5. mv

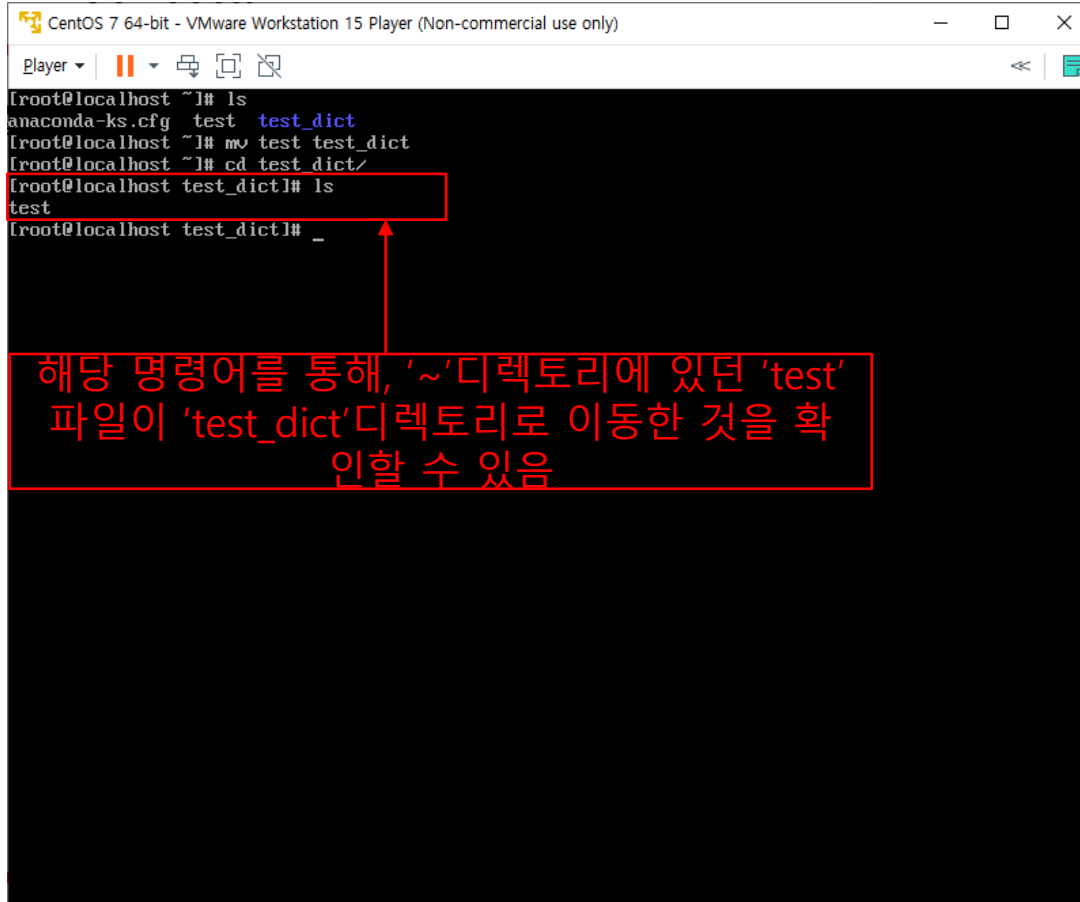
- ✓ mv는 특정 파일을 다른 디렉토리로 이동시키는 명령어임
- ✓ 'mv 파일명 디렉토리경로'와 같은 형식으로 입력하면 됨
- ✓ '디렉토리경로'를 기입할 때, 상대경로 또는 절대경로로 기입하면 됨



```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
root@localhost ~]# ls
anaconda-ks.cfg test test_dict
root@localhost ~]# mv test test_dict
```

A-5. mv

✓ 'mv test test_dict' 명령어를 실행한 결과



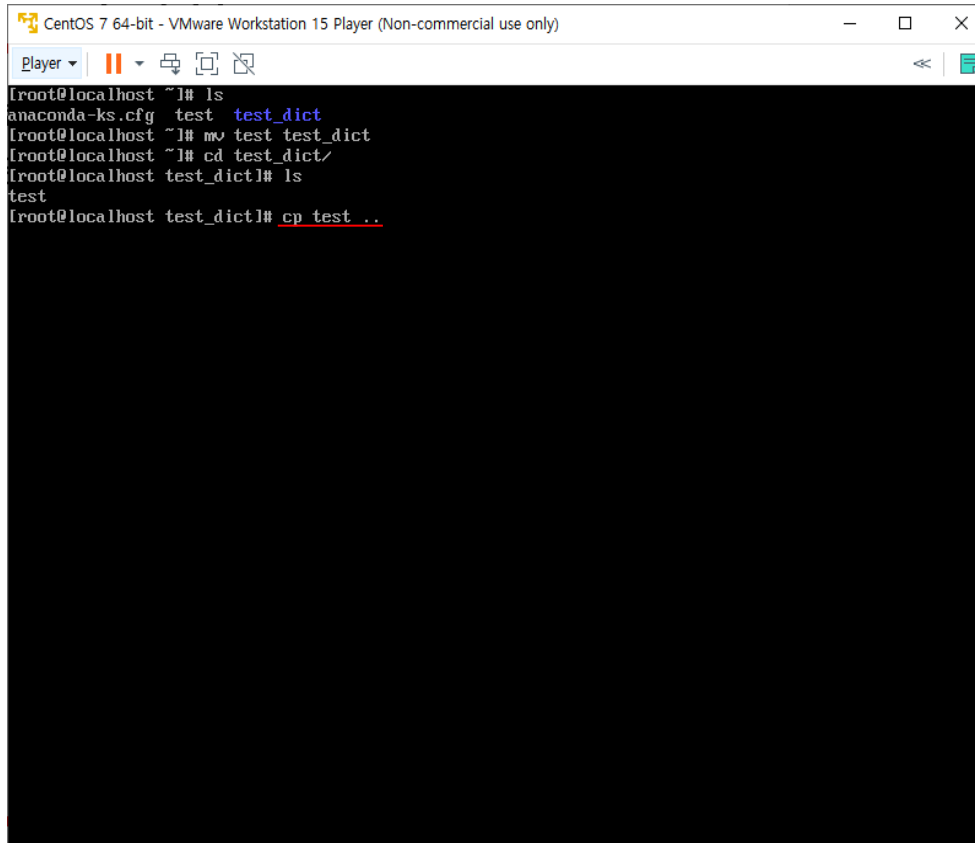
```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player | [Icons]
[root@localhost ~]# ls
anaconda-ks.cfg  test  test_dict
[root@localhost ~]# mv test test_dict
[root@localhost ~]# cd test_dict/
[root@localhost test_dict]# ls
test
[root@localhost test_dict]# _
```

해당 명령어를 통해, '~'디렉토리에 있던 'test' 파일이 'test_dict'디렉토리로 이동한 것을 확인할 수 있음

❖ 참고로, 해당 명령어를 실행하면 'test'파일은 더 이상 '~'디렉토리에 남아있지 않음

A-6. cp

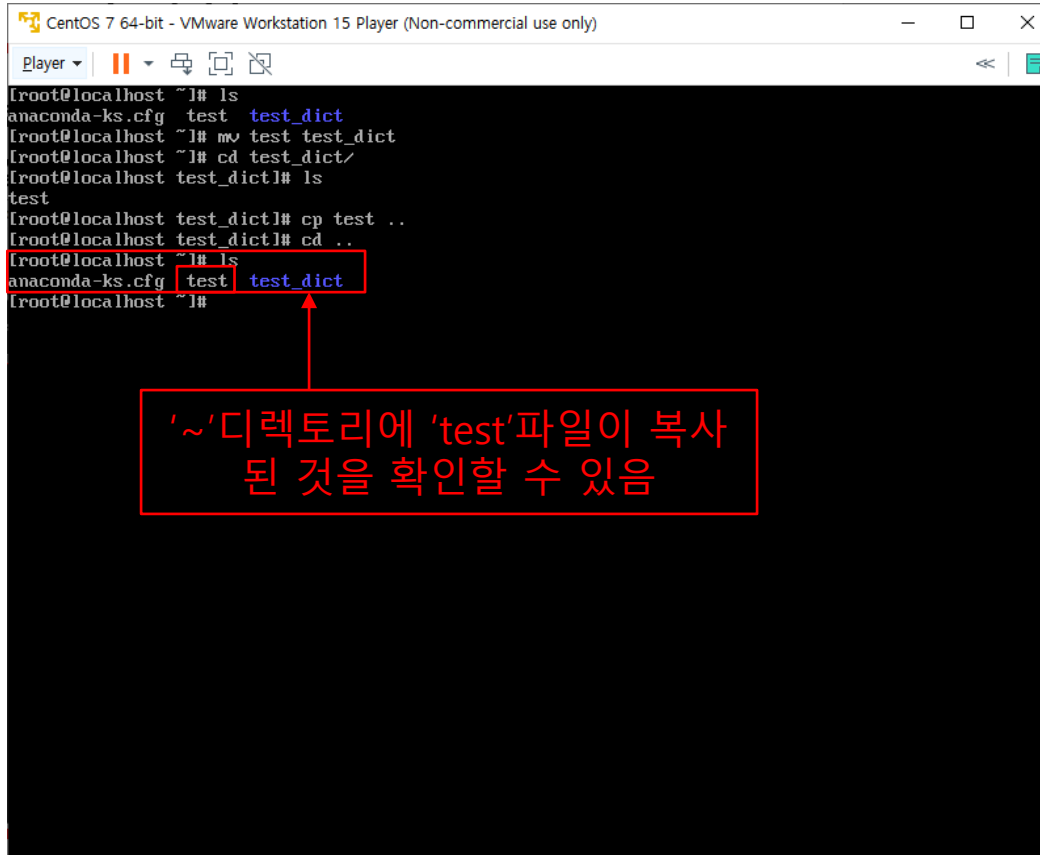
- ✓ cp는 특정 파일을 다른 디렉토리 내에 복사하는 명령어임
- ✓ 'cp 파일명 디렉토리경로'와 같은 형식으로 입력하면 됨
- ✓ '디렉토리경로'를 기입할 때, 상대경로 또는 절대경로로 기입하면 됨



```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
[root@localhost ~]# ls
anaconda-ks.cfg  test  test_dict
[root@localhost ~]# mv test test_dict
[root@localhost ~]# cd test_dict/
[root@localhost test_dict]# ls
test
[root@localhost test_dict]# cp test ..
```


A-6. cp

✓ 'cp test ..'명령어를 실행한 결과



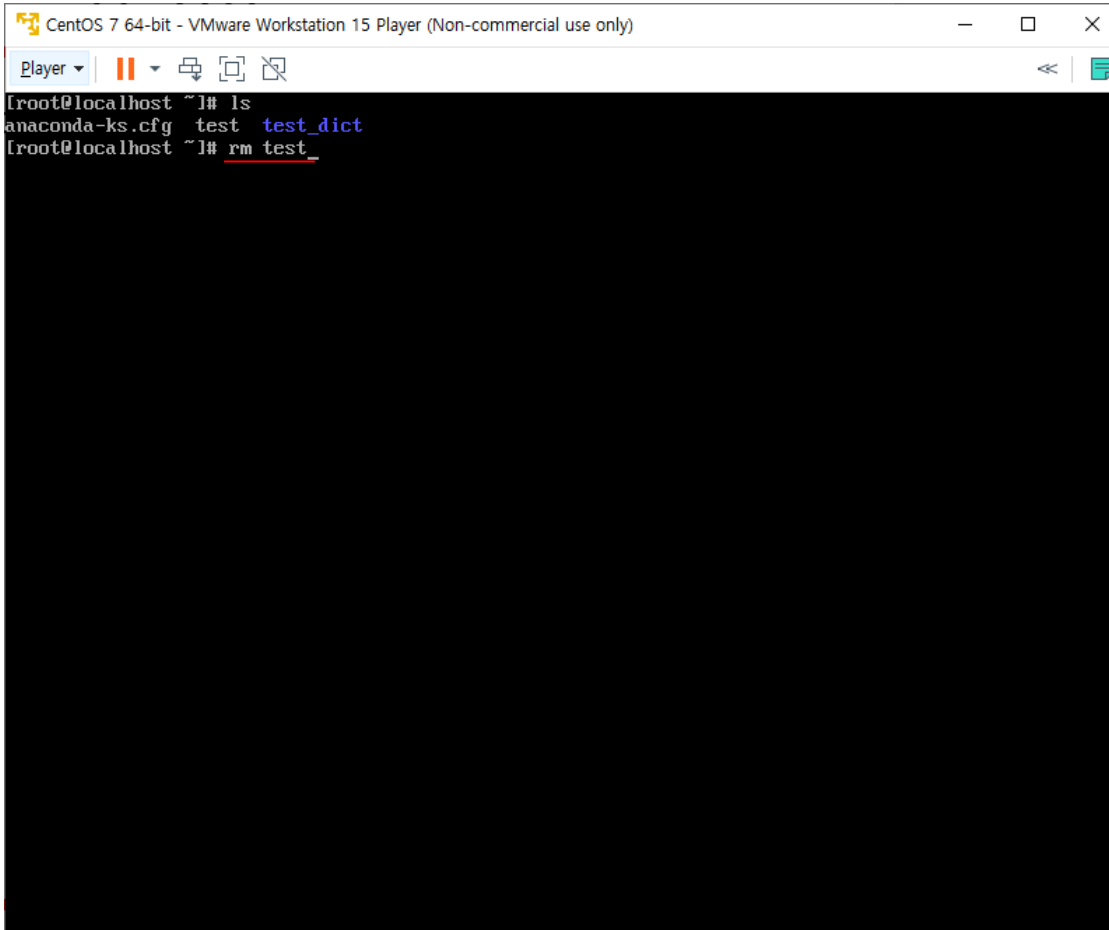
```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
[root@localhost ~]# ls
anaconda-ks.cfg  test  test_dict
[root@localhost ~]# mv test test_dict
[root@localhost ~]# cd test_dict/
[root@localhost test_dict]# ls
test
[root@localhost test_dict]# cp test ..
[root@localhost test_dict]# cd ..
[root@localhost ~]# ls
anaconda-ks.cfg  test  test_dict
[root@localhost ~]#
```

'~'디렉토리에 'test'파일이 복사
된 것을 확인할 수 있음

❖ 참고로, 해당 명령어를 실행하면 'test'파일은 '~'디렉토리와 'test_dict'디렉토리에 모두 존재함

A-7. rm

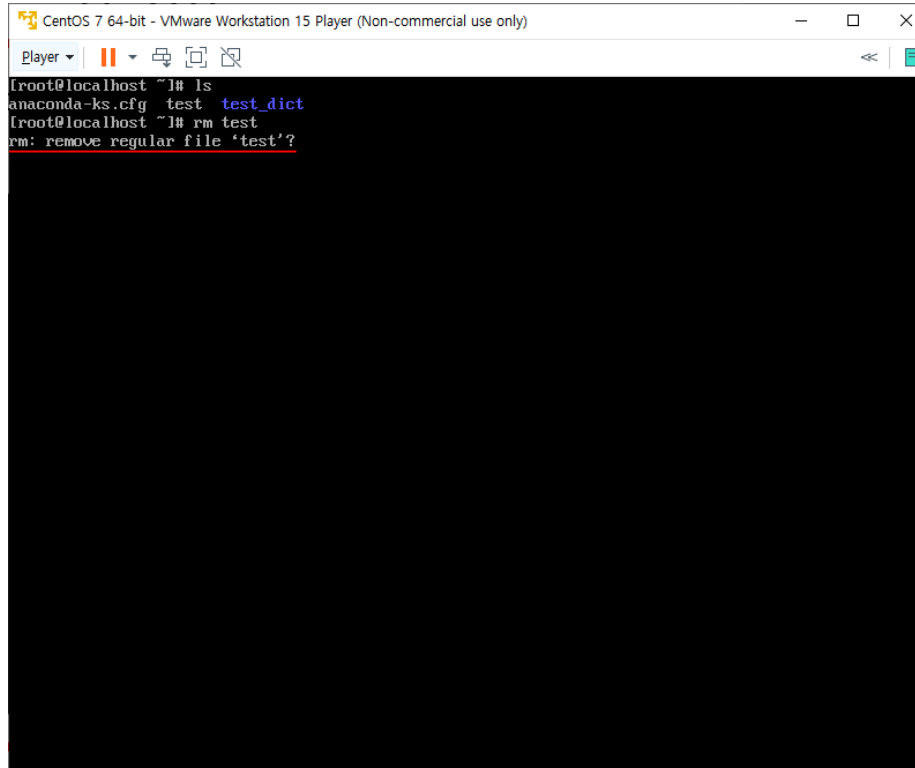
- ✓ rm은 사용자가 현재 위치한 디렉토리 내 특정 파일을 삭제하는 명령어임



```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
[root@localhost ~]# ls
anaconda-ks.cfg  test  test_dict
[root@localhost ~]# rm test
```

A-7. rm

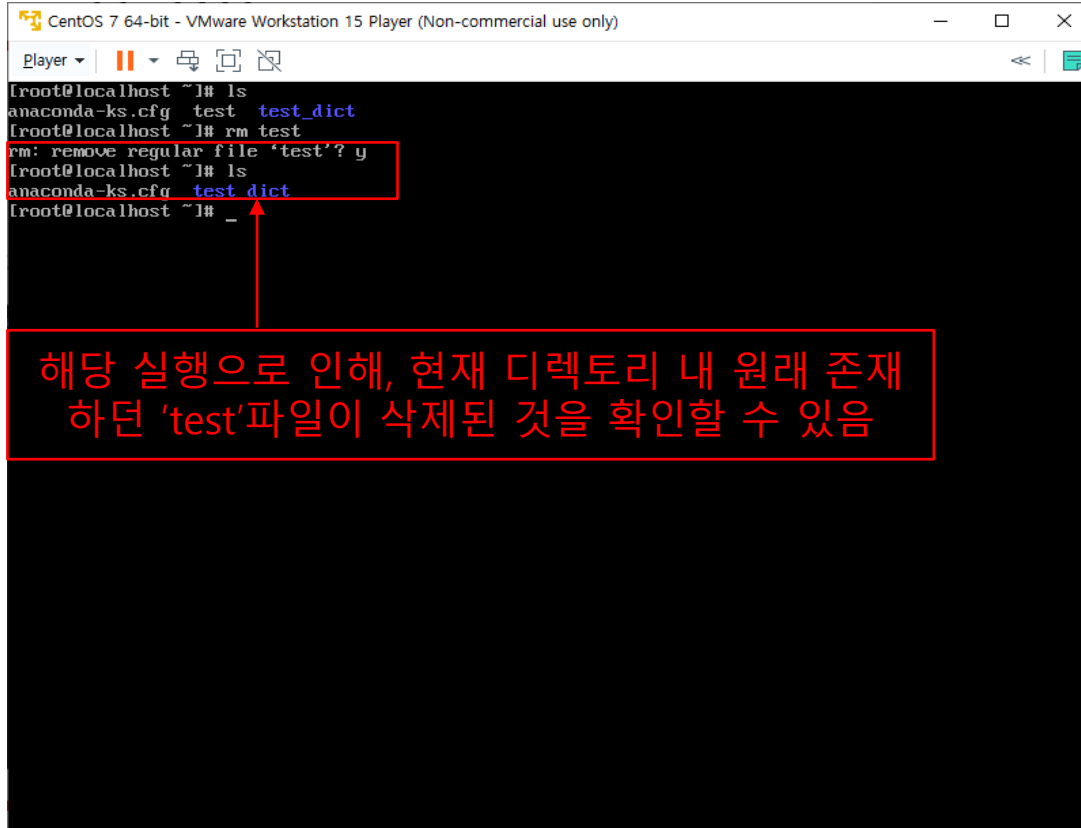
- ✓ 'rm test'명령어를 실행하면, 실제로 해당 파일을 지울 것인지 물어보는 메시지가 뜬다
- ✓ 이 상황에서 'y'를 입력하고 enter키를 누르면 해당 파일이 삭제되고, 'n'을 입력하고 enter키를 누르면 해당 파일이 삭제되지 않음



```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
[root@localhost ~]# ls
anaconda-ks.cfg  test  test_dict
[root@localhost ~]# rm test
rm: remove regular file 'test'?
```

A-7. rm

✓ 'y'를 입력 후 enter키를 누른 결과



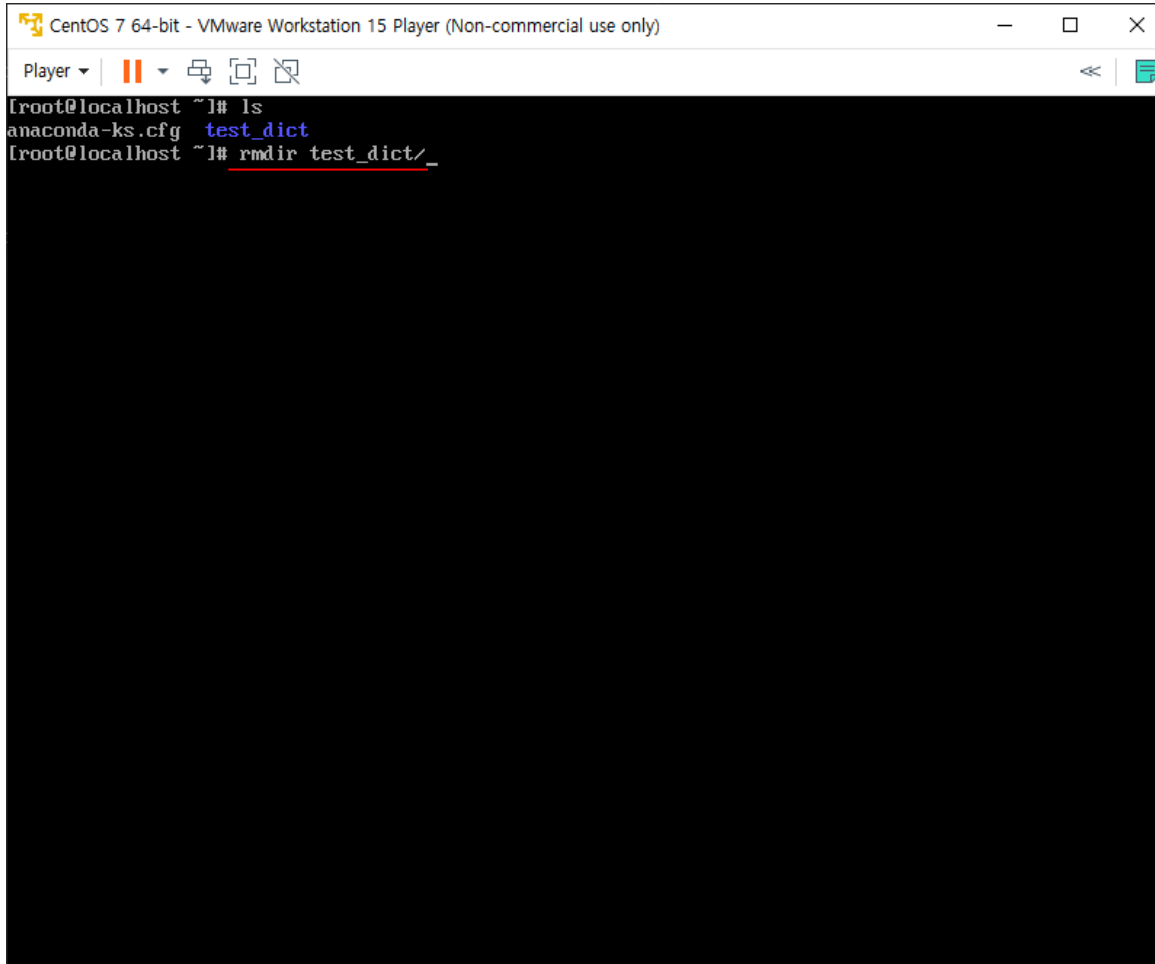
```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player | [Icons]
[root@localhost ~]# ls
anaconda-ks.cfg  test  test_dict
[root@localhost ~]# rm test
rm: remove regular file 'test'? y
[root@localhost ~]# ls
anaconda-ks.cfg  test_dict
[root@localhost ~]#
```

해당 실행으로 인해, 현재 디렉토리 내 원래 존재 하던 'test'파일이 삭제된 것을 확인할 수 있음

❖ 참고로, 해당 파일을 실제로 지울 것인지 물어보는 메시지를 띄우지 않고 파일을 삭제하려면, 'rm -f'명령어를 입력하면 됨

A-8. rmdir

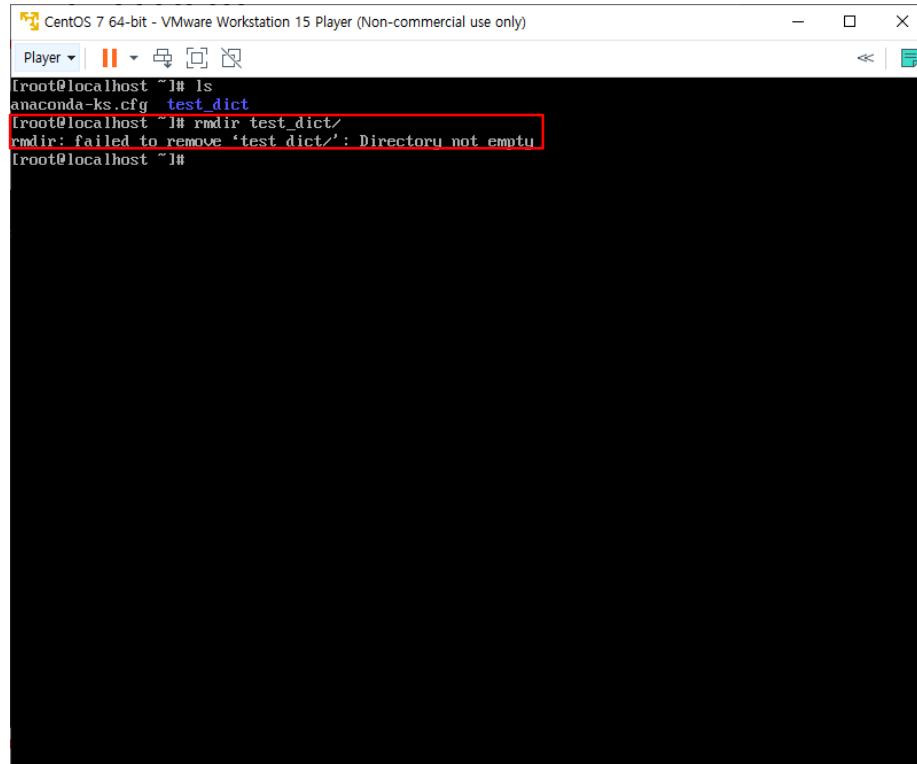
- ✓ rmdir은 사용자가 위치한 디렉토리 내 특정 디렉토리를 삭제하는 명령어임



```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player | [Icons]
[root@localhost ~]# ls
anaconda-ks.cfg  test_dict
[root@localhost ~]# rmdir test_dict/
```

A-8. rmdir

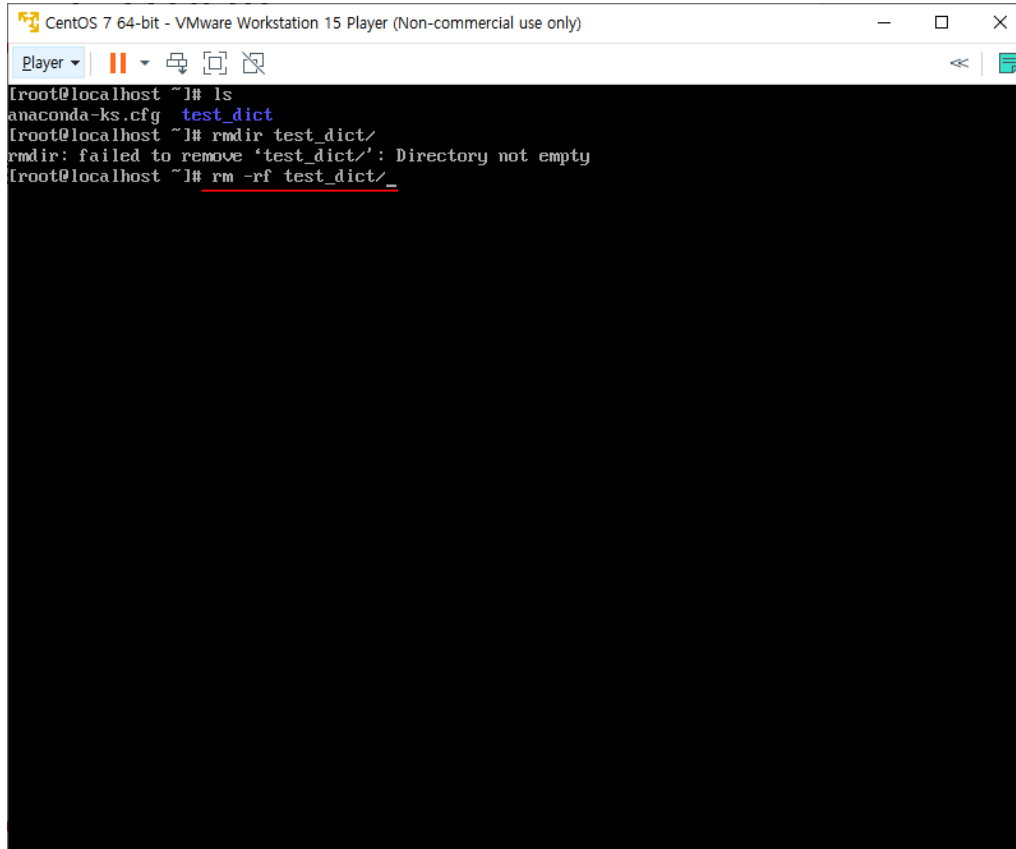
- ✓ 'rmdir test_dict' 명령어를 실행한 결과로, '해당 디렉토리를 삭제하는 것이 실패됨'이라는 메시지가 뜬다
- ✓ 'test_dict' 디렉토리가 비어 있지 않기 때문에 삭제할 수 없다는 메시지가 뜨는 것임(해당 디렉토리에 'test' 파일이 존재함)



```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
[root@localhost ~]# ls
anaconda-ks.cfg  test_dict
[root@localhost ~]# rmdir test_dict/
rmdir: failed to remove 'test_dict/': Directory not empty
[root@localhost ~]#
```

A-8. rmdir

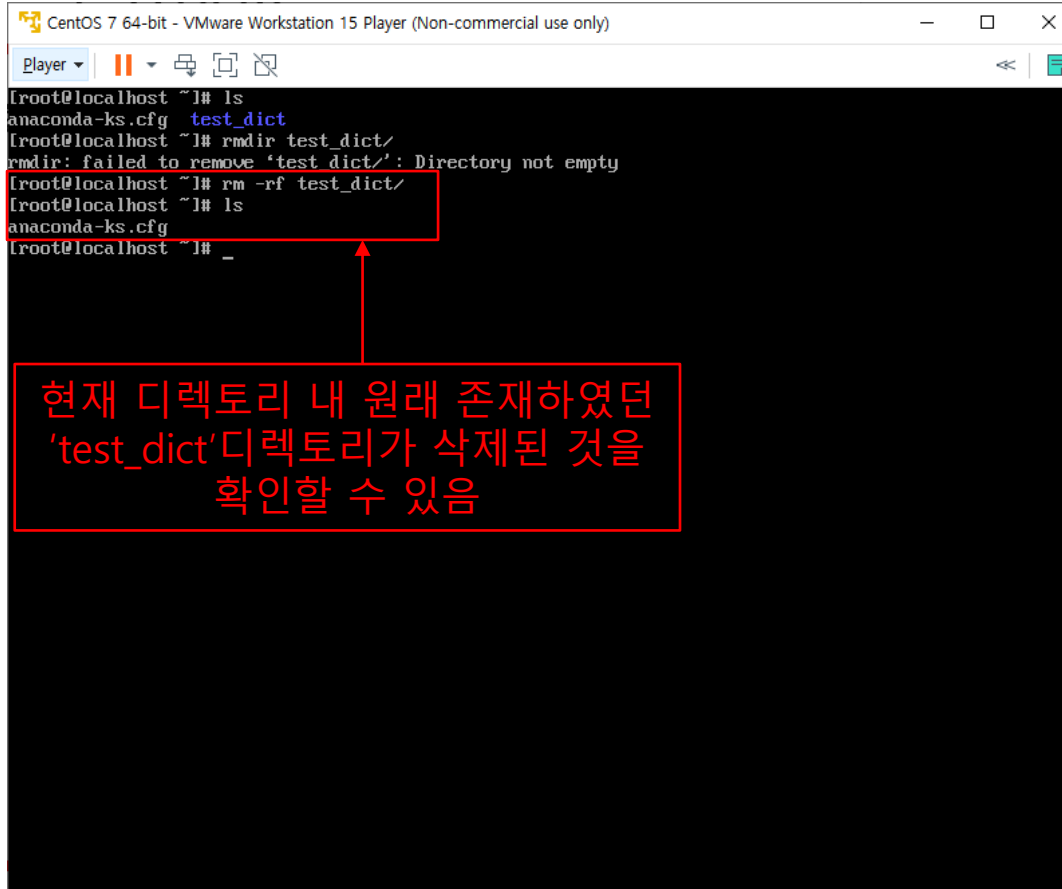
- ✓ 해당 메시지를 무시하고 강제로 비어 있지 않은 디렉토리를 지우기 위해선, 'rm -rf 디렉토리명'과 같은 명령어를 입력하면 됨



```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
[root@localhost ~]# ls
anaconda-ks.cfg  test_dict
[root@localhost ~]# rmdir test_dict/
rmdir: failed to remove 'test_dict/': Directory not empty
[root@localhost ~]# rm -rf test_dict/
```

A-8. rmdir

✓ 'rm -rf test_dict'명령어를 실행한 결과

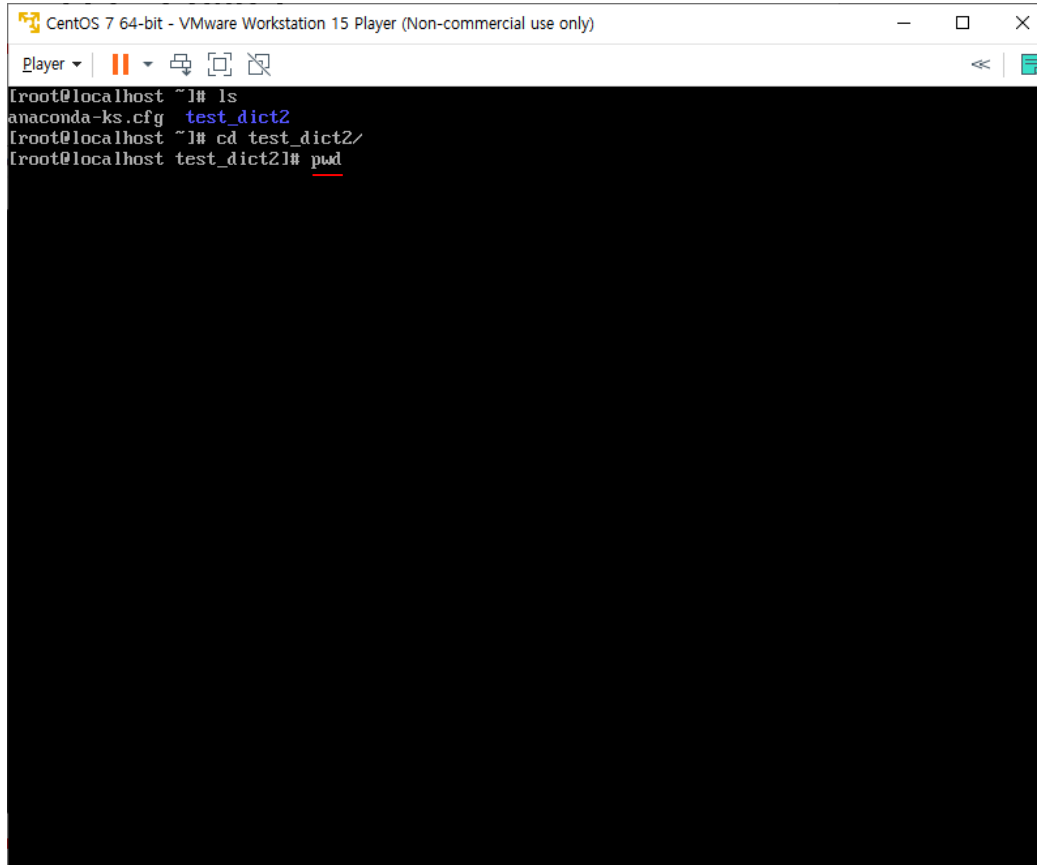


```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
root@localhost ~]# ls
anaconda-ks.cfg test_dict
root@localhost ~]# rmdir test_dict/
rmdir: failed to remove 'test_dict/': Directory not empty
root@localhost ~]# rm -rf test_dict/
root@localhost ~]# ls
anaconda-ks.cfg
root@localhost ~]# _
```

현재 디렉토리 내 원래 존재하였던 'test_dict'디렉토리가 삭제된 것을 확인할 수 있음

A-9. pwd

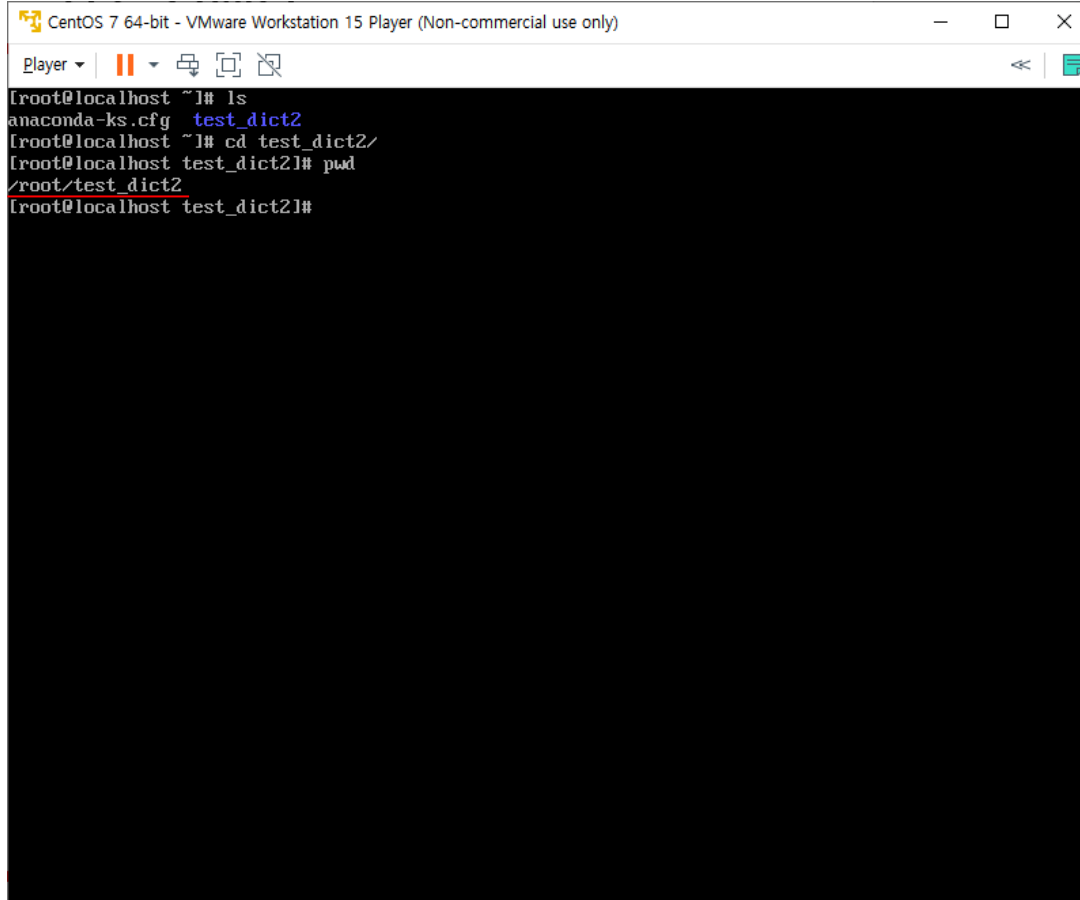
- ✓ pwd는 사용자가 현재 위치한 디렉토리의 절대경로를 보여주는 명령어임
- ✓ 밑의 예시의 상황은 '~'디렉토리에서 'test_dict2'디렉토리를 생성하고, 'test_dict2'디렉토리로 이동함



```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
root@localhost ~]# ls
anaconda-ks.cfg test_dict2
root@localhost ~]# cd test_dict2/
root@localhost test_dict2]# pwd
```

A-9. pwd

✓ 'pwd'명령어를 실행한 결과



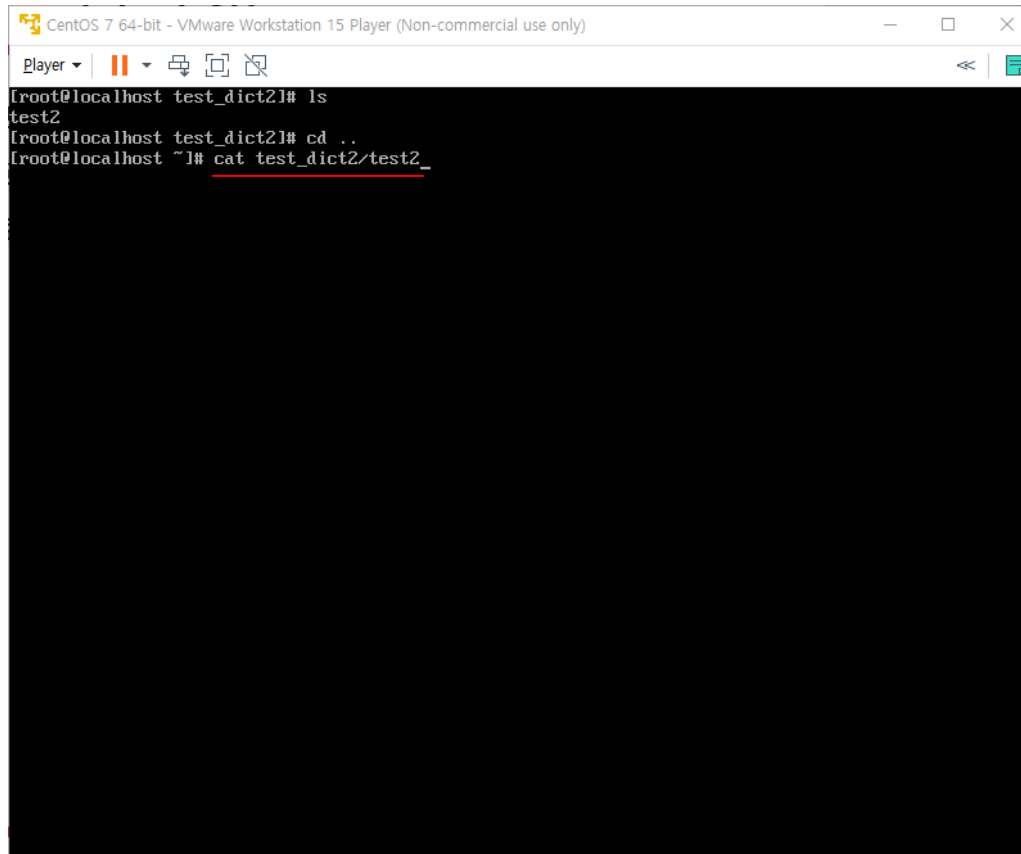
```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
[root@localhost ~]# ls
anaconda-ks.cfg  test_dict2
[root@localhost ~]# cd test_dict2/
[root@localhost test_dict2]# pwd
/root/test_dict2
[root@localhost test_dict2]#
```

A-10. cat

- ✓ cat은 특정 파일의 내용물을 보여주는 명령어임
- ✓ 'cat 파일경로'와 같은 형식으로 입력하면 됨
- ✓ '파일경로'를 입력할 때는 상대경로 또는 절대경로로 입력하면 됨

A-10. cat

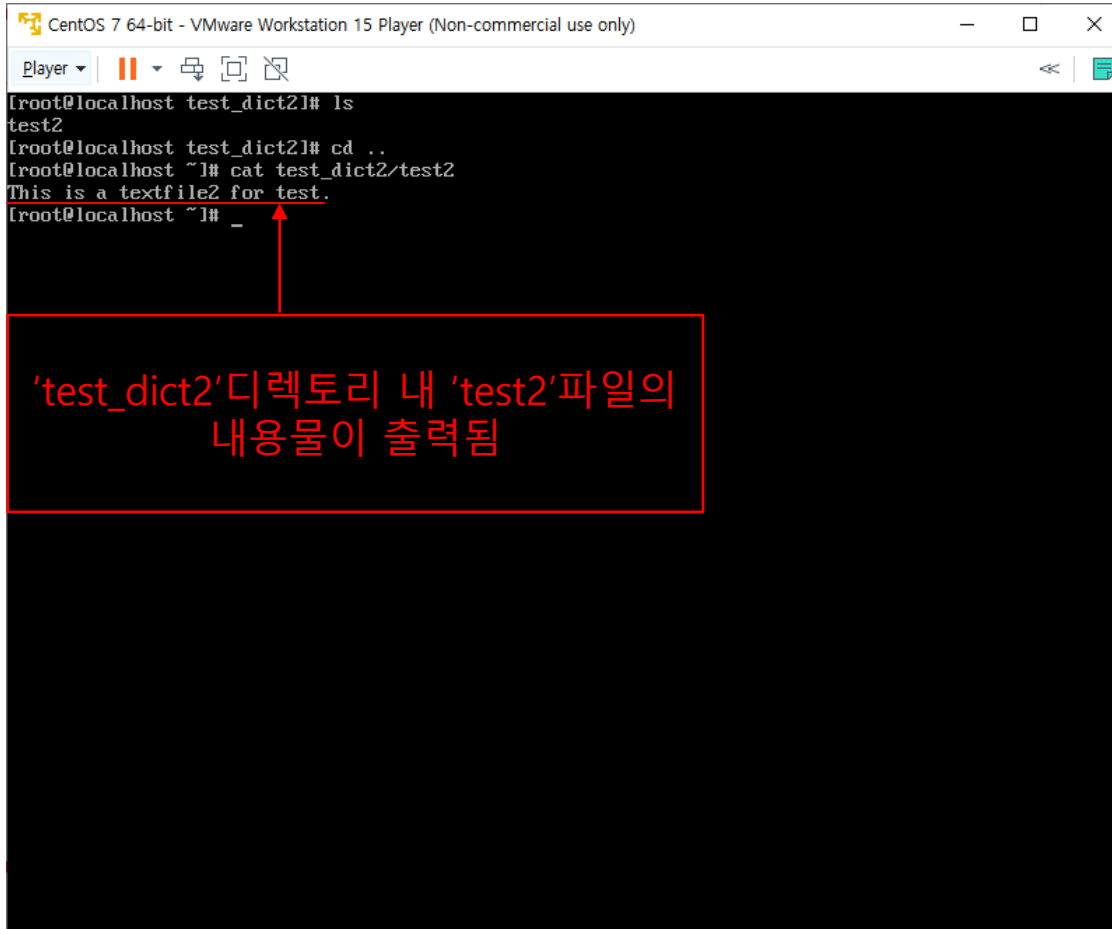
- ✓ 밑의 예시는 'test_dict2'디렉토리에 test2라는 파일을 생성하고, '~'디렉토리로 이동한 상황임



```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player | [Icons]
[root@localhost test_dict2]# ls
test2
[root@localhost test_dict2]# cd ..
[root@localhost ~]# cat test_dict2/test2_
```

A-10. cat

✓ 'cat test_dict2/test2' 명령어를 실행한 결과

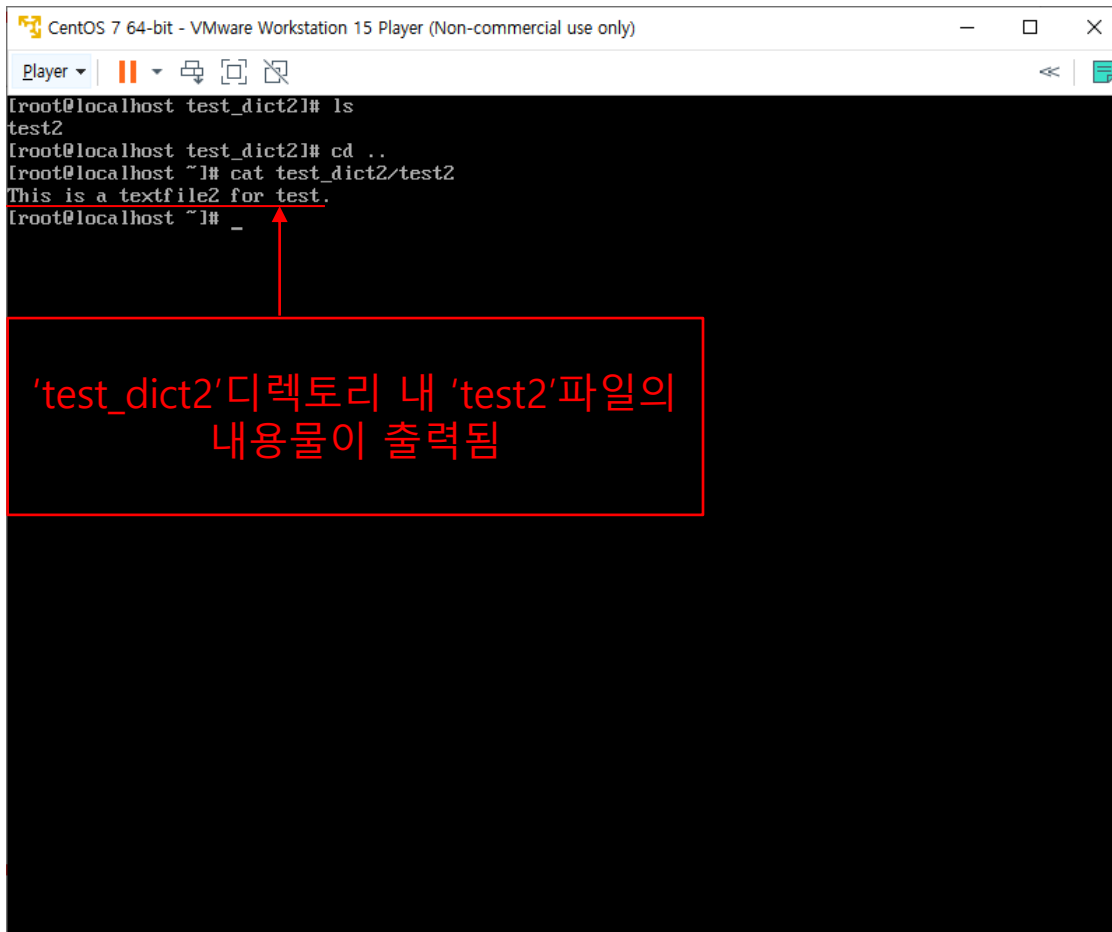


```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
[root@localhost test_dict2]# ls
test2
[root@localhost test_dict2]# cd ..
[root@localhost ~]# cat test_dict2/test2
This is a textfile2 for test.
[root@localhost ~]#
```

'test_dict2' 디렉토리 내 'test2' 파일의
내용물이 출력됨

A-10. cat

✓ 'cat test_dict2/test2' 명령어를 실행한 결과



```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
[root@localhost test_dict2]# ls
test2
[root@localhost test_dict2]# cd ..
[root@localhost ~]# cat test_dict2/test2
This is a textfile2 for test.
[root@localhost ~]#
```

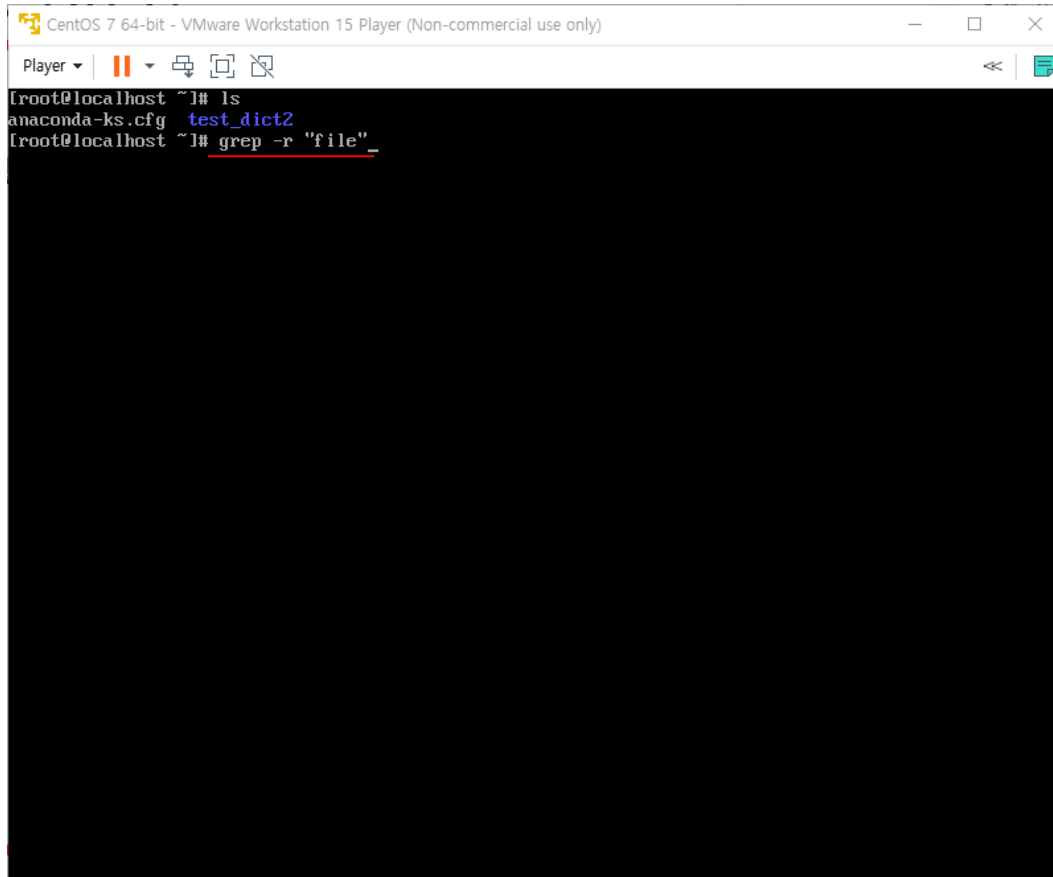
'test_dict2' 디렉토리 내 'test2' 파일의
내용물이 출력됨

A-11. grep

- ✓ grep은 입력으로 전달된 파일의 내용에서 특정 문자열을 찾고자 할 때 사용하는 명령어임
- ✓ grep는 보통 '-r' 옵션과 함께 사용함
- ✓ grep -r은 현재 사용자가 위치한 디렉토리 내 하위 디렉토리들까지 검사하여, 해당 문자열이 어느 파일의 어떤 문장에 적혀 있는지 보여줌
- ✓ 'grep -r 문자열'과 같은 형식으로 입력하면 됨

A-11. grep

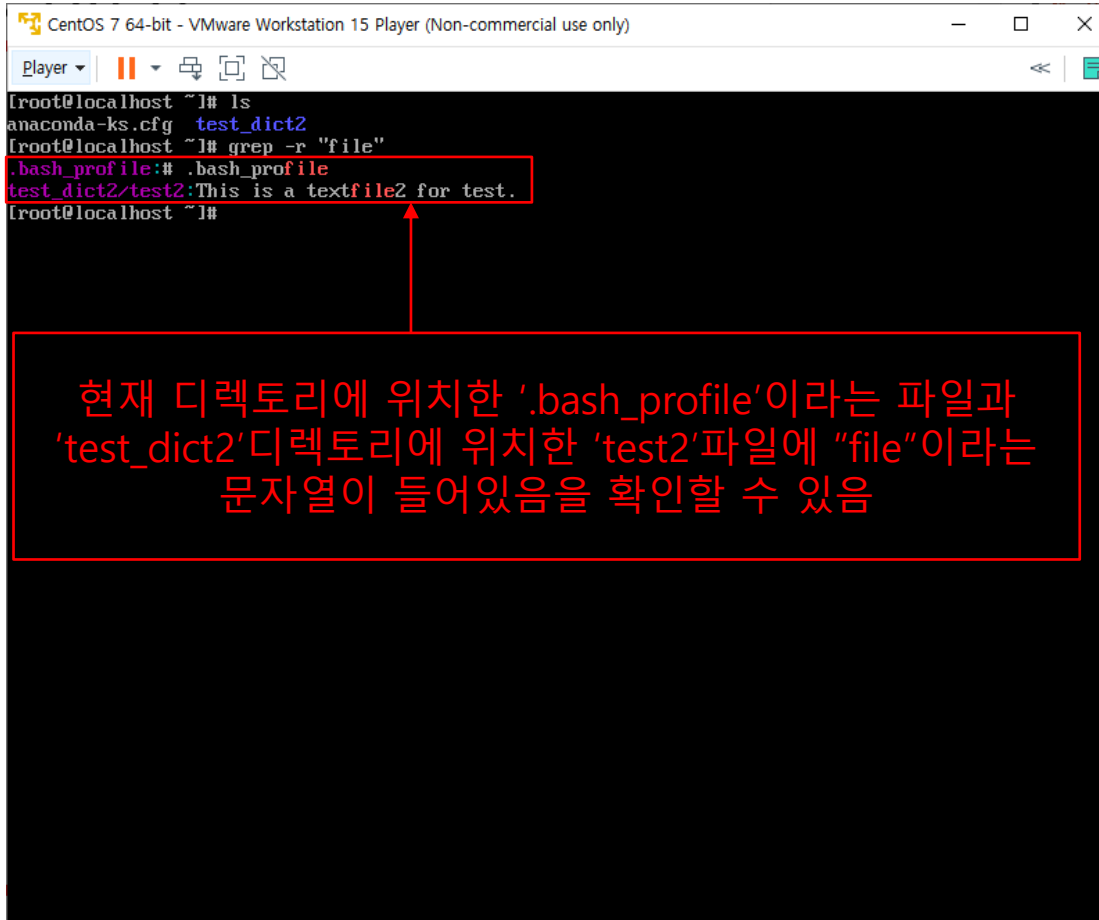
- ✓ 밑에 예시는 '~' 디렉토리에서 하위 디렉토리까지 검사하여, "file"이라는 문자열이 어느 파일의 어떤 문장에 포함되어 있는지 찾으려고 하는 상황임



```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
[root@localhost ~]# ls
anaconda-ks.cfg  test_dict2
[root@localhost ~]# grep -r "file"
```


A-11. grep

✓ 'grep -r "file"' 명령어를 실행한 결과

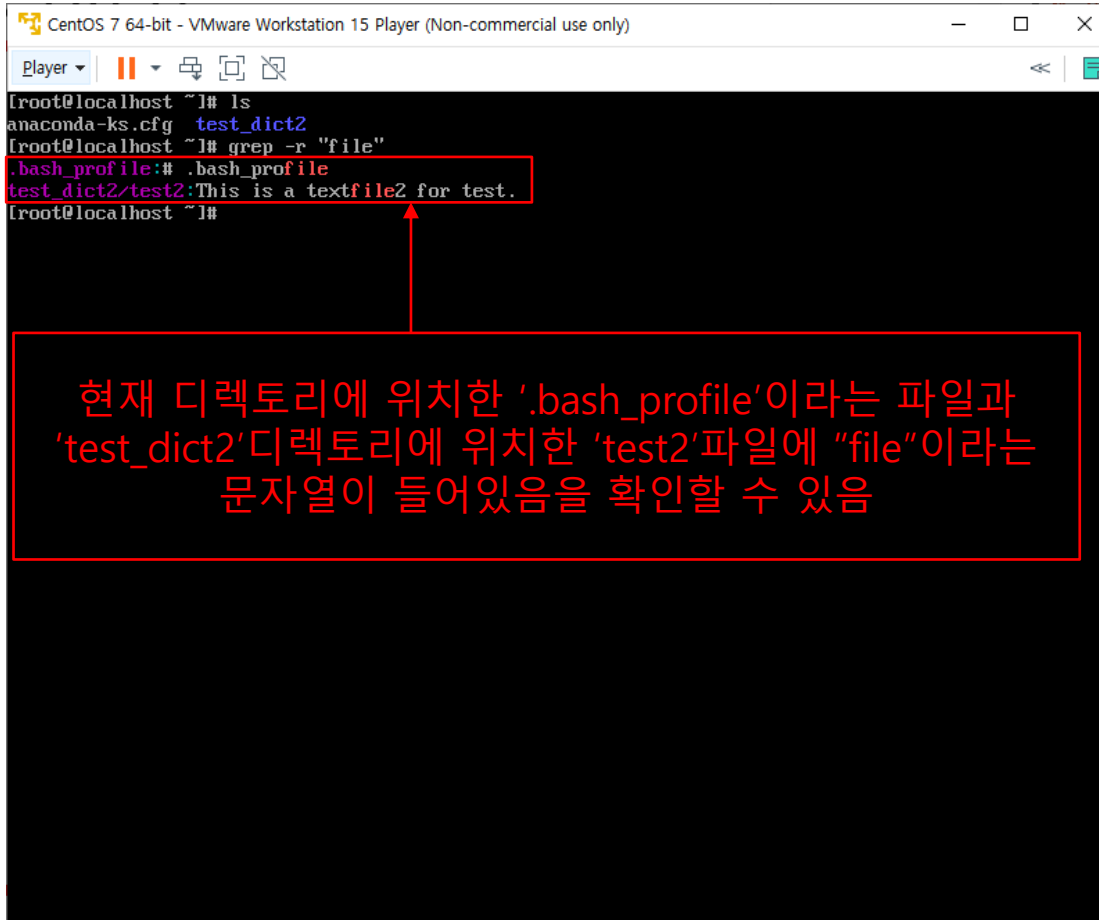


```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
[root@localhost ~]# ls
anaconda-ks.cfg  test_dict2
[root@localhost ~]# grep -r "file"
.bash_profile:# .bash_profile
test_dict2/test2:This is a textfile2 for test.
[root@localhost ~]#
```

현재 디렉토리에 위치한 '.bash_profile'이라는 파일과
'test_dict2' 디렉토리에 위치한 'test2' 파일에 "file"이라는
문자열이 들어있음을 확인할 수 있음

A-11. grep

✓ 'grep -r "file"' 명령어를 실행한 결과

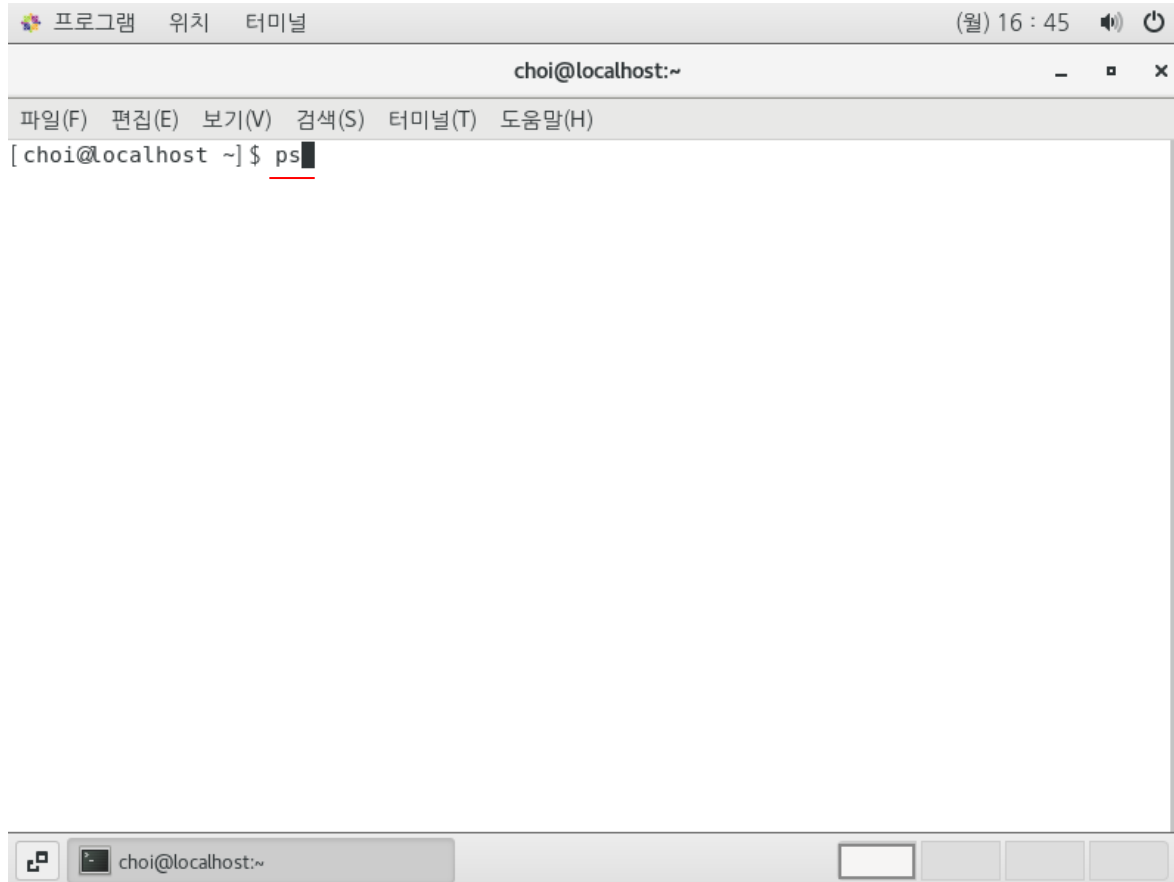


```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
[root@localhost ~]# ls
anaconda-ks.cfg  test_dict2
[root@localhost ~]# grep -r "file"
.bash_profile:# .bash_profile
test_dict2/test2:This is a textfile2 for test.
[root@localhost ~]#
```

현재 디렉토리에 위치한 '.bash_profile'이라는 파일과
'test_dict2' 디렉토리에 위치한 'test2' 파일에 "file"이라는
문자열이 들어있음을 확인할 수 있음

A-12. ps

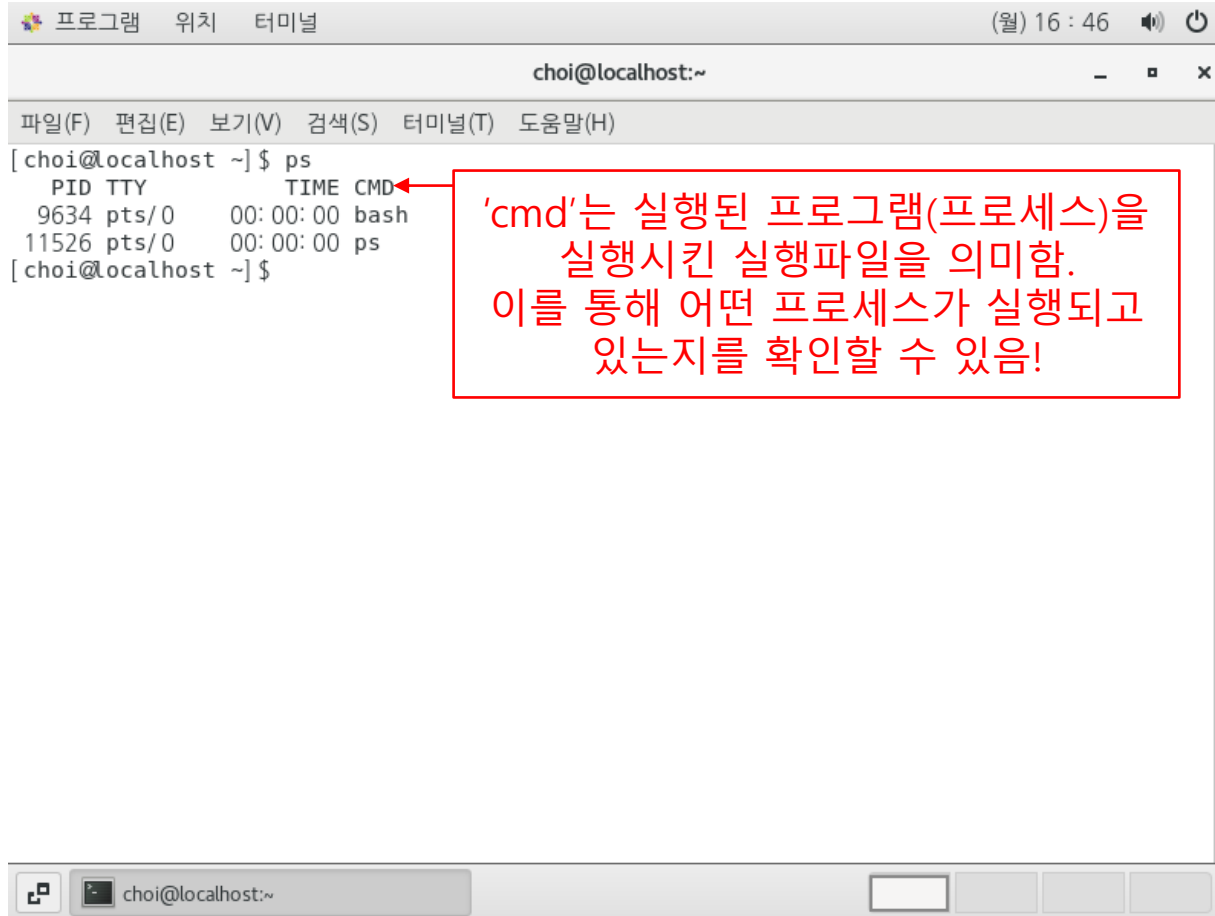
- ✓ ps는 현재 실행된 프로그램(프로세스)의 목록을 보여주는 명령어임
- ✓ '-e', '-f', '-l', '-p', '-u' 등의 옵션들이 존재하고, 이 중 '-e'와 '-u'이 주로 사용됨



The screenshot shows a terminal window titled "프로그램 위치 터미널" with a timestamp of "(월) 16 : 45". The prompt is "choi@localhost:~". The menu bar includes "파일(F)", "편집(E)", "보기(V)", "검색(S)", "터미널(T)", and "도움말(H)". The command prompt shows "[choi@localhost ~] \$" followed by the command "ps" which is underlined. The terminal is otherwise empty.

A-12. ps

✓ ps명령어 실행 결과

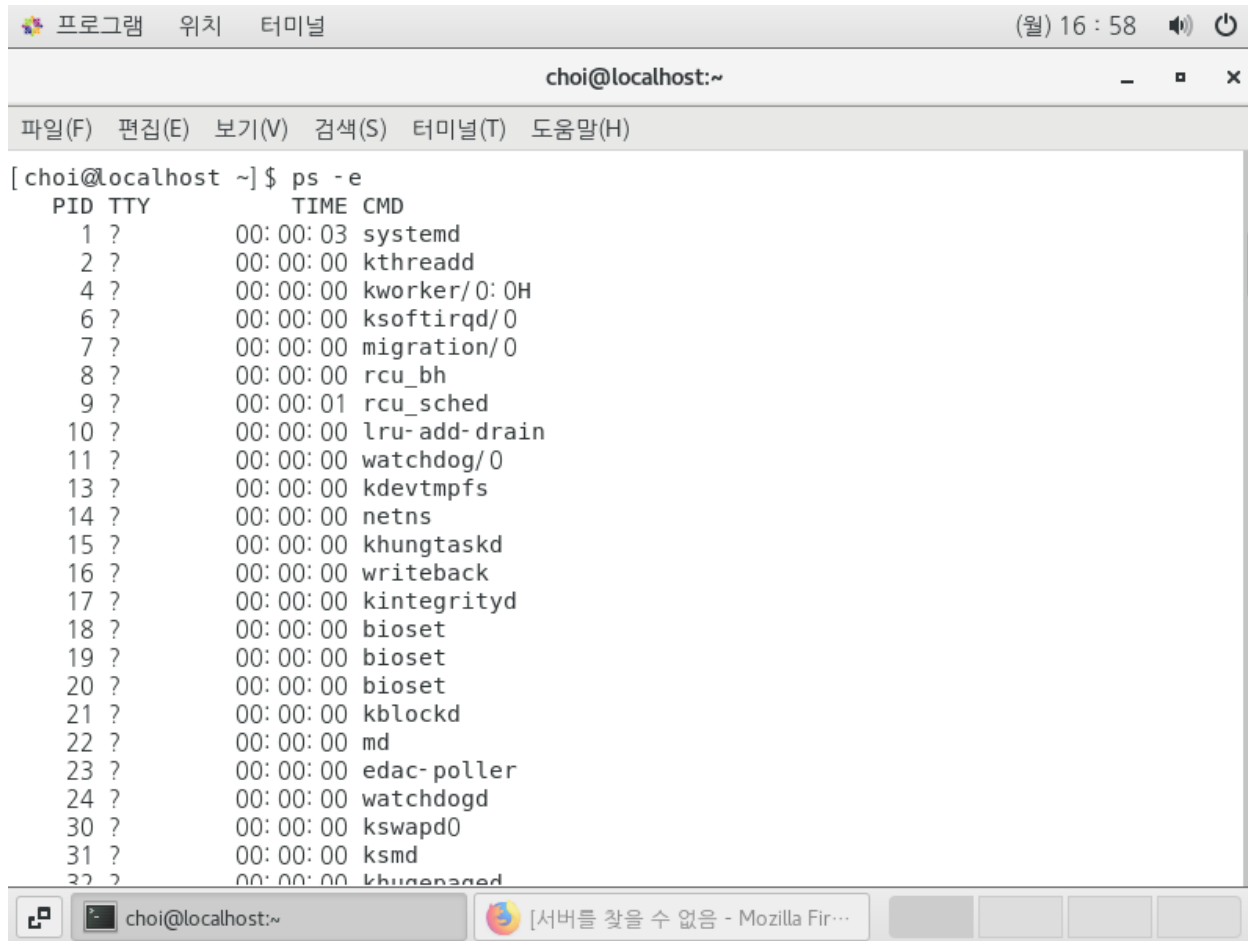


```
[choi@localhost ~]$ ps
  PID TTY          TIME CMD
  9634 pts/0    00:00:00 bash
 11526 pts/0    00:00:00 ps
[choi@localhost ~]$
```

'cmd'는 실행된 프로그램(프로세스)을
실행시킨 실행파일을 의미함.
이를 통해 어떤 프로세스가 실행되고
있는지를 확인할 수 있음!

A-12. ps

- ✓ ps-e명령어는 현재의 모든 프로세스를 나타냄



```
[choi@localhost ~]$ ps -e
```

PID	TTY	TIME	CMD
1	?	00:00:03	systemd
2	?	00:00:00	kthreadd
4	?	00:00:00	kworker/0:0H
6	?	00:00:00	ksoftirqd/0
7	?	00:00:00	migration/0
8	?	00:00:00	rcu_bh
9	?	00:00:01	rcu_sched
10	?	00:00:00	lru-add-drain
11	?	00:00:00	watchdog/0
13	?	00:00:00	kdevtmpfs
14	?	00:00:00	netns
15	?	00:00:00	khungtaskd
16	?	00:00:00	writeback
17	?	00:00:00	kintegrityd
18	?	00:00:00	bioaset
19	?	00:00:00	bioaset
20	?	00:00:00	bioaset
21	?	00:00:00	kblockd
22	?	00:00:00	md
23	?	00:00:00	edac-poller
24	?	00:00:00	watchdogd
30	?	00:00:00	kswapd0
31	?	00:00:00	ksmd
32	?	00:00:00	khugepaged

A-12. ps

- ✓ ps -u 명령어는 특정 사용자의 프로세스를 나타냄
- ✓ 'ps -u 사용자명'과 같은 형식으로 입력하면 됨

```

[choi@localhost ~]$ ps -u choi
  PID TTY          TIME CMD
  4038 ?        00:00:00 gnome-keyring-d
  4917 ?        00:00:00 gnome-session-b
  4950 ?        00:00:00 dbus-launch
  4955 ?        00:00:00 dbus-daemon
  5036 ?        00:00:00 imsettings-daem
  5046 ?        00:00:00 gvfsd
  5075 ?        00:00:00 gvfsd-fuse
  5268 ?        00:00:00 ssh-agent
  5332 ?        00:00:00 at-spi-bus-laun
  5343 ?        00:00:00 dbus-daemon
  5353 ?        00:00:00 at-spi2-registr
  5430 ?        00:00:24 gnome-shell
  5490 ?        00:00:01 pulseaudio
  5612 ?        00:00:00 ibus-daemon
  5623 ?        00:00:00 ibus-dconf
  5630 ?        00:00:00 ibus-x11
  5638 ?        00:00:00 ibus-portal
  5678 ?        00:00:00 xdg-permission-
  5685 ?        00:00:00 gnome-shell-cal
  5708 ?        00:00:00 evolution-sourc
  5736 ?        00:00:00 dconf-service
  5745 ?        00:00:00 goa-daemon
  5769 ?        00:00:00 mission-control
  5777 ?        00:00:00 gupc-audio2-us
  
```

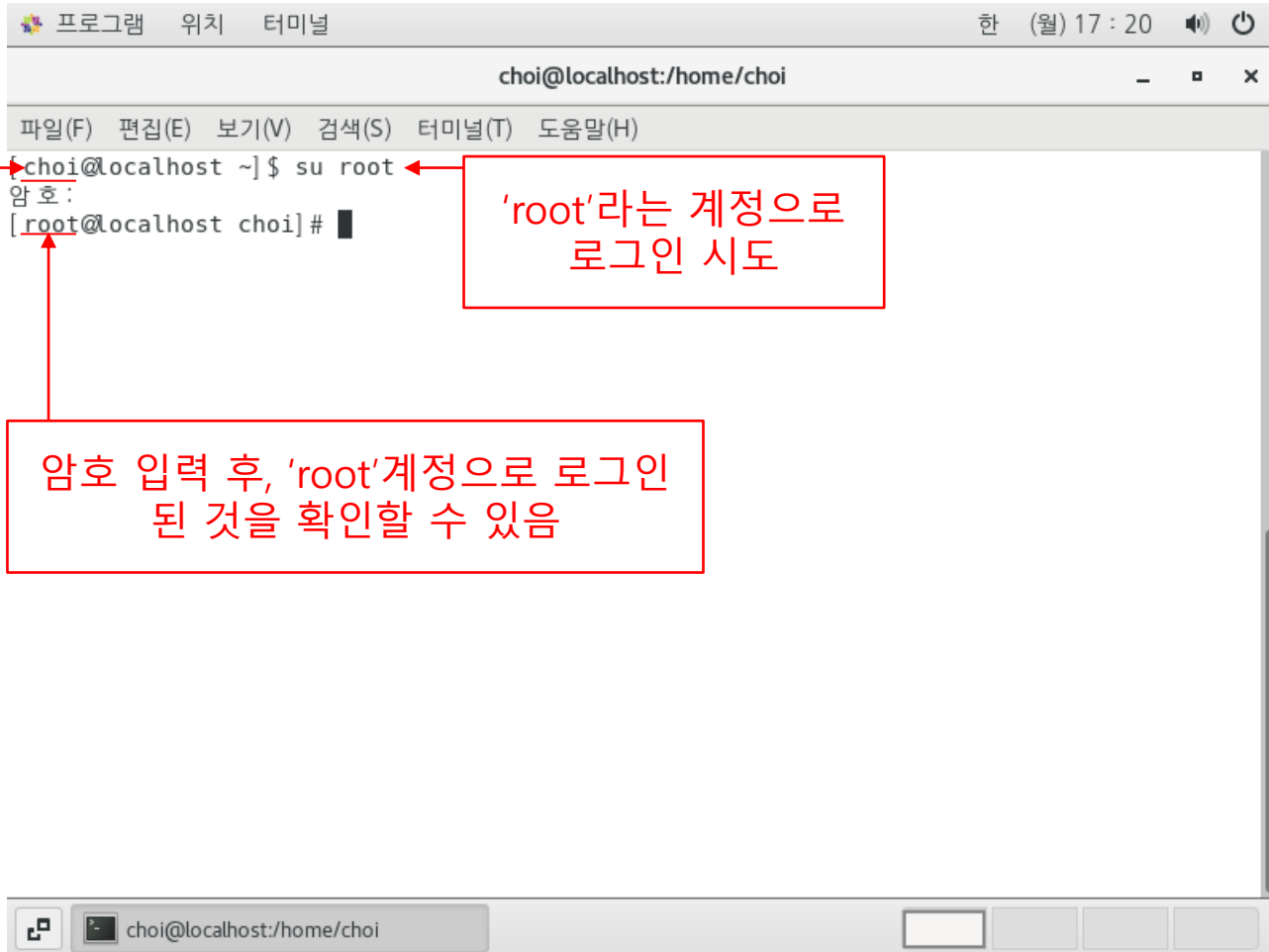
'choi'라는 사용자가 실행한 프로세스를 나타냄

A-13. su

- ✓ su는 현재 사용자가 사용하고 있는 계정에서 다른 계정으로 전환시키는 명령어임
- ✓ Linux OS에는 해당 OS를 사용하는 여러 계정이 생성될 수 있고, 사용자들은 이 중 한 개의 계정에 로그인하여 해당 OS를 사용하고 있는 것임(Window OS도 동일함)
- ✓ Linux OS는 기본적으로 'root'라는 계정을 가지고 있음
- ✓ 즉, 사용자가 Linux OS를 설치할 때 특정 계정을 하나 생성한다면, 해당 Linux OS에는 root라는 계정과 방금 생성한 계정 두 개가 존재하는 것임

A-13. su

- ✓ su 명령어를 통해, 원래 사용하고 있던 계정에서 다른 계정으로 로그인하는 모습



```
choi@localhost:~/home/choi
[choi@localhost ~]$ su root
암호:
[root@localhost choi]#
```

원래 사용하고 있던 계정

'root'라는 계정으로 로그인 시도

암호 입력 후, 'root'계정으로 로그인 된 것을 확인할 수 있음

A-14. chmod

- ✓ chmod는 파일과 디렉토리의 권한을 변경해주는 명령어임
- ✓ 'chmod 권한 파일'의 형식과 같이 입력하면 됨
- ✓ 예를 들어 test라는 파일에 다른 사용자의 쓰기 권한을 추가하고 싶다면,
'chmod o+w test'라고 입력하면 됨
- ✓ 그리고 test라는 파일에 사용자의 쓰기 권한을 제거하고 싶다면,
'chmod u-w test'라고 입력하면 됨
- ✓ 또한 test라는 파일에 사용자와 그룹의 읽기, 쓰기 권한을 추가하고 싶다면,
'chmod ug+rw test'라고 입력하면 됨

A-14. chmod

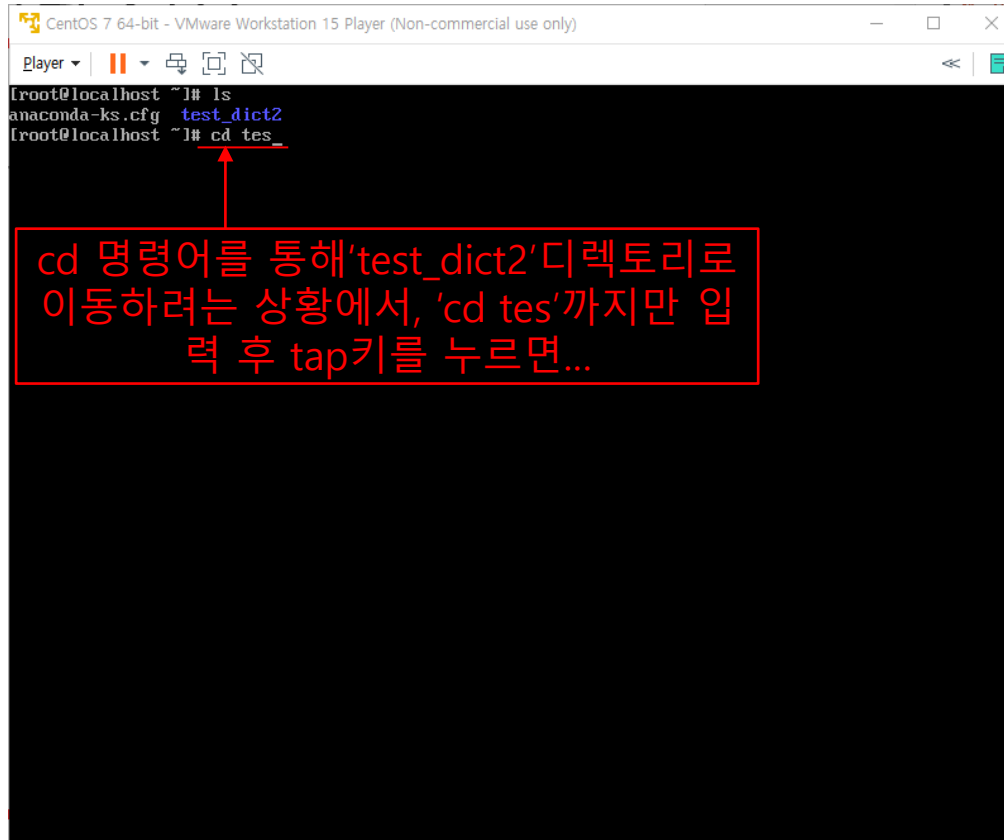
✓ 'test'파일에 다른 사용자의 쓰기 권한을 추가하는 예시

```

[choi@localhost ~]$ ls -l
합계 4
-rw-rw-r--. 1 choi choi 5  5월 11 19:12 test
drwxr-xr-x. 2 choi choi 6  5월  9 23:05 공개
drwxr-xr-x. 2 choi choi 6  5월  9 23:05 다운로드
drwxr-xr-x. 2 choi choi 6  5월  9 23:05 문서
drwxr-xr-x. 2 choi choi 6  5월  9 23:05 바탕화면
drwxr-xr-x. 2 choi choi 6  5월  9 23:05 비디오
drwxr-xr-x. 2 choi choi 6  5월  9 23:05 사진
drwxr-xr-x. 2 choi choi 6  5월  9 23:05 서식
drwxr-xr-x. 2 choi choi 6  5월  9 23:05 음악
[choi@localhost ~]$ chmod o+w test
[choi@localhost ~]$ ls -l
합계 4
-rw-rw-rw-. 1 choi choi 5  5월 11 19:12 test
drwxr-xr-x. 2 choi choi 6  5월  9 23:05 공개
drwxr-xr-x. 2 choi choi 6  5월  9 23:05 다운로드
drwxr-xr-x. 2 choi choi 6  5월  9 23:05 문서
drwxr-xr-x. 2 choi choi 6  5월  9 23:05 바탕화면
drwxr-xr-x. 2 choi choi 6  5월  9 23:05 비디오
drwxr-xr-x. 2 choi choi 6  5월  9 23:05 사진
drwxr-xr-x. 2 choi choi 6  5월  9 23:05 서식
drwxr-xr-x. 2 choi choi 6  5월  9 23:05 음악
[choi@localhost ~]$
  
```

A-15. 기타 기능

- ✓ 특정 명령어를 입력하는 상황에서 파일 경로 또는 디렉토리 경로를 입력해야 할 때, tap키를 누르면 해당 경로를 자동완성 할 수 있음

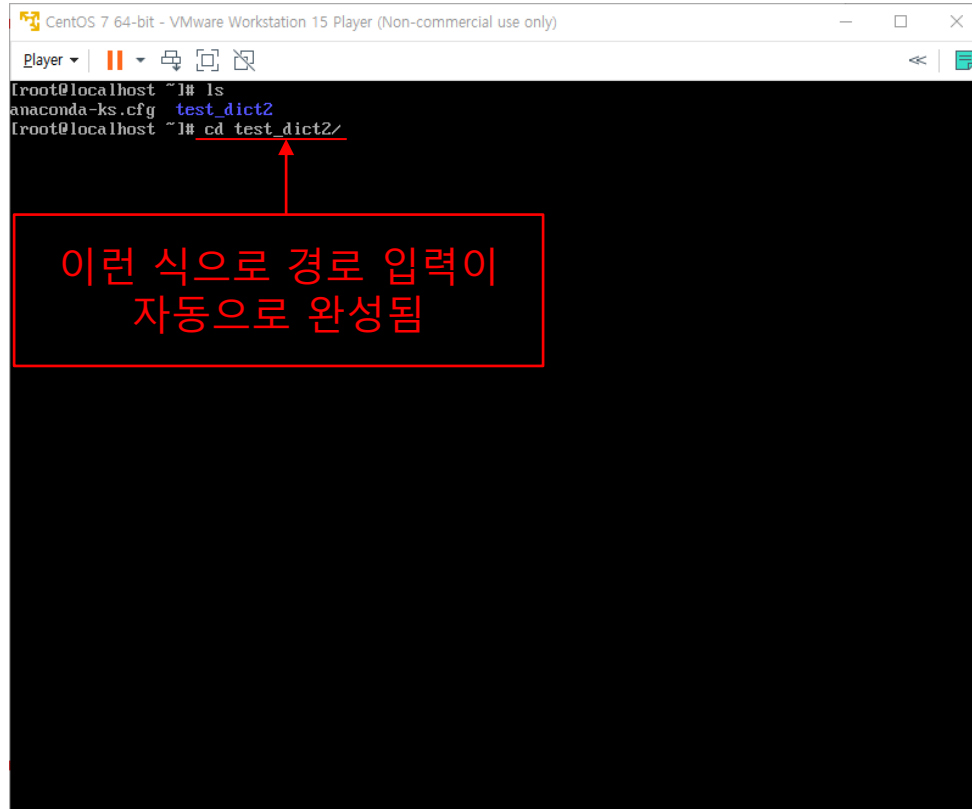


```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player | [Icons]
[root@localhost ~]# ls
anaconda-ks.cfg test_dict2
[root@localhost ~]# cd tes_
```

cd 명령어를 통해 'test_dict2' 디렉토리로 이동하려는 상황에서, 'cd tes'까지만 입력 후 tap키를 누르면...

A-15. 기타 기능

✓ tap키를 누른 결과

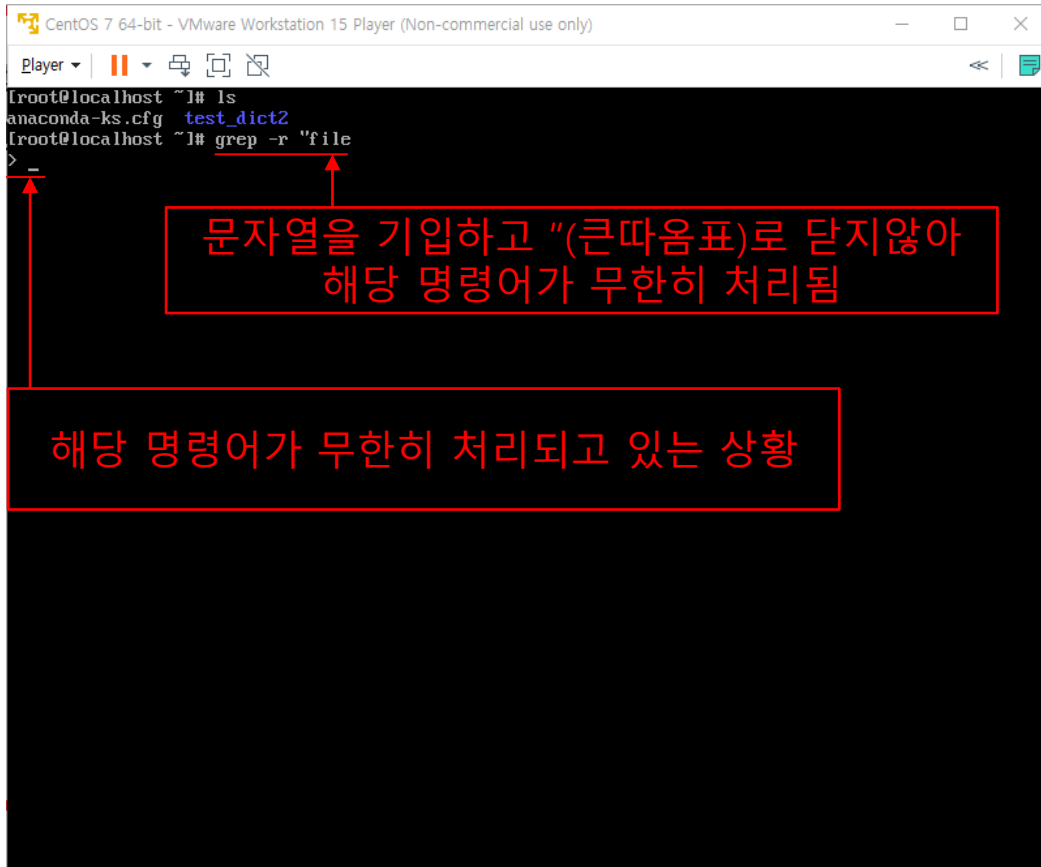


```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
root@localhost ~]# ls
anaconda-ks.cfg test_dict2
root@localhost ~]# cd test_dict2/
```

이런 식으로 경로 입력이
자동으로 완성됨

A-15. 기타 기능

- ✓ 만약 특정 명령어를 잘못 입력하여 Linux가 해당 명령어를 무한히 처리하는 상황이 발생한다면, 'ctrl + c'키를 누르면 이 명령어 처리가 중단됨



The screenshot shows a terminal window titled "CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)". The terminal displays the following commands and output:

```
root@localhost ~]# ls
anaconda-ks.cfg test_dict2
root@localhost ~]# grep -r "file
>
```

A red box with Korean text is overlaid on the terminal, pointing to the command line:

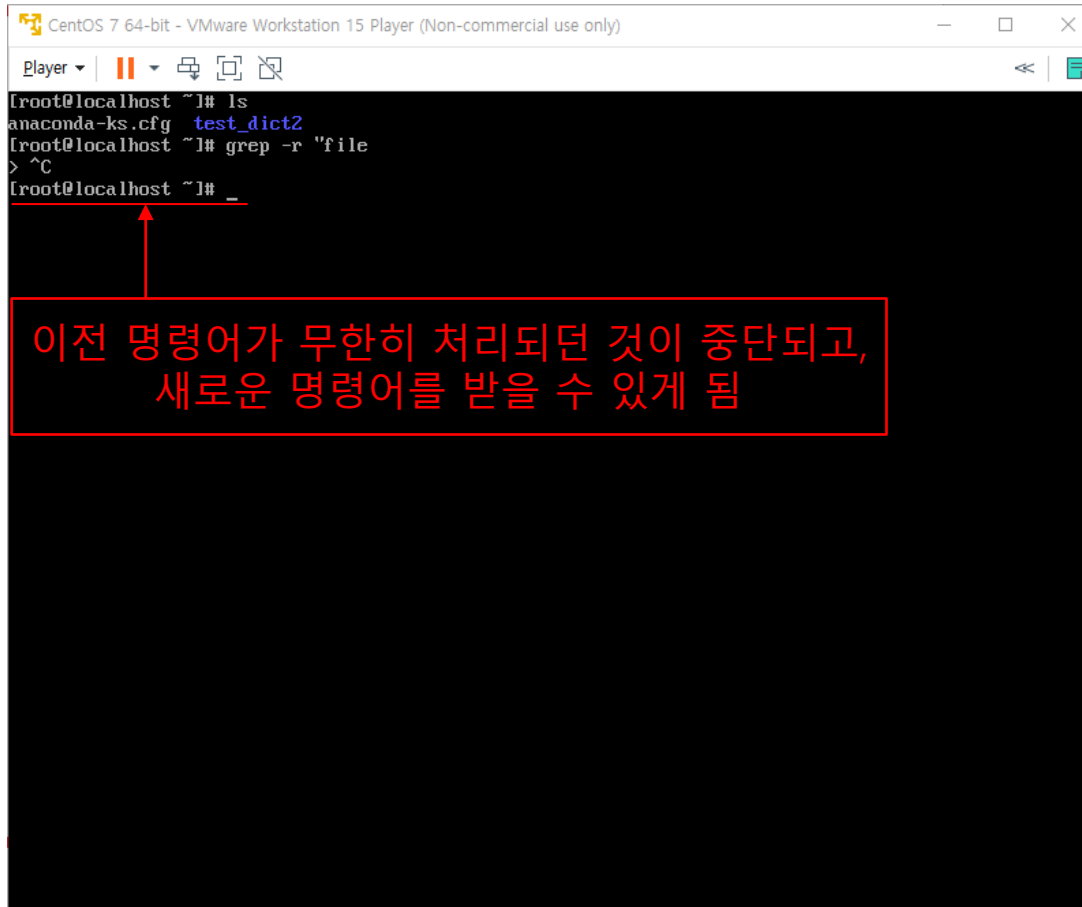
문자열을 기입하고 "(큰따옴표)로 닫지않아
해당 명령어가 무한히 처리됨

Another red box with Korean text is overlaid on the terminal, pointing to the prompt:

해당 명령어가 무한히 처리되고 있는 상황

A-15. 기타 기능

✓ 'ctrl + c'키를 누른 결과

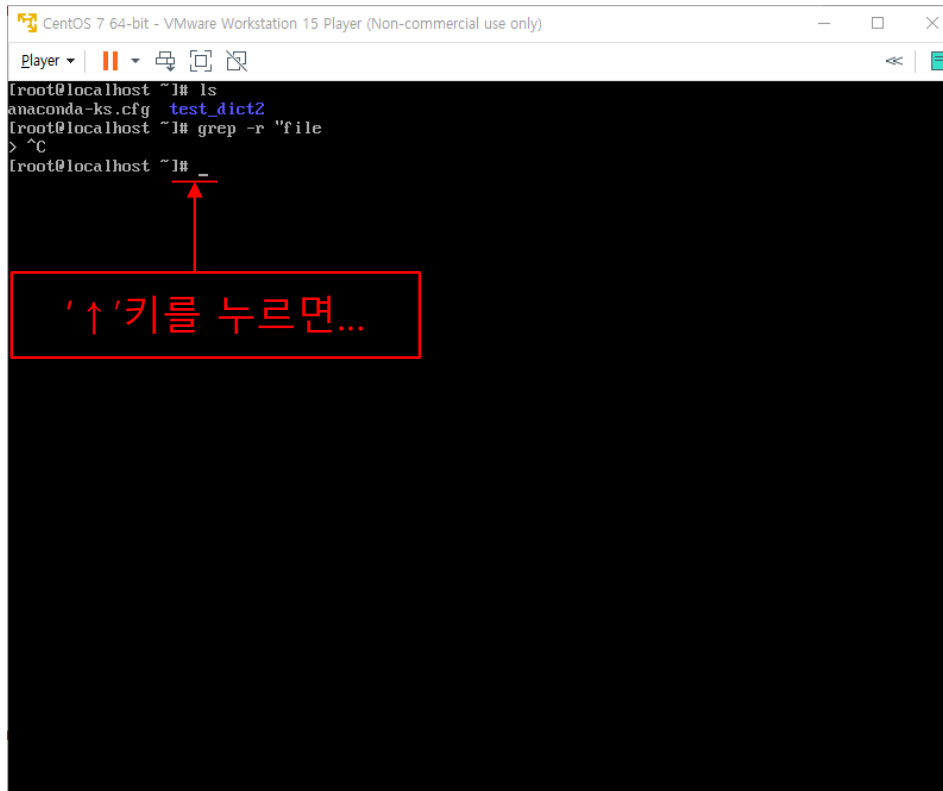


```
CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player | [Pause] [Full Screen] [Close]
[root@localhost ~]# ls
anaconda-ks.cfg  test_dict2
[root@localhost ~]# grep -r "file"
> ^C
[root@localhost ~]# _
```

이전 명령어가 무한히 처리되던 것이 중단되고,
새로운 명령어를 받을 수 있게 됨

A-15. 기타 기능

- ✓ 명령어를 입력할 때, 사용자가 이전에 입력했던 명령어들을 다시 불러오고 시다면 '↑'키를 누르면 됨
- ✓ 해당 키를 누르면, 최근에 입력했던 명령어들 순으로 명령어가 자동완성 됨



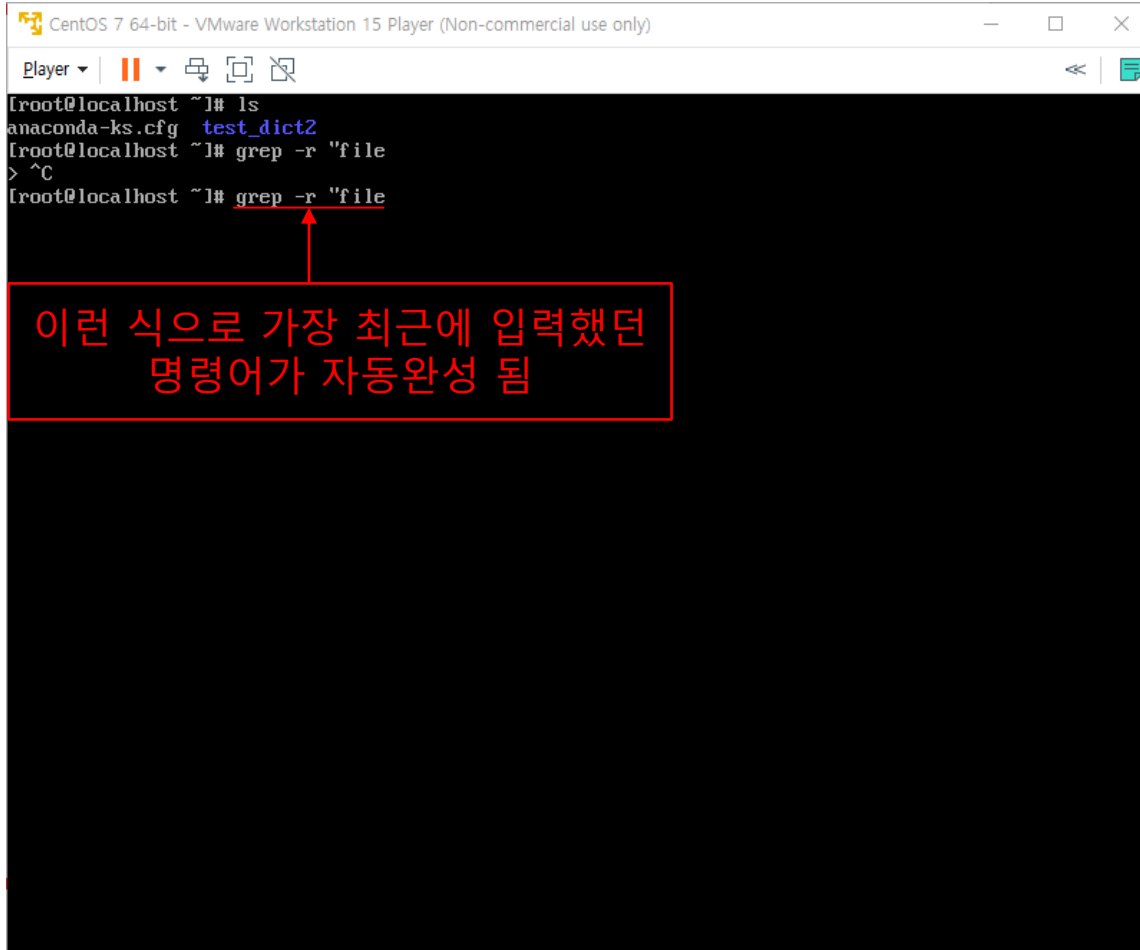
The screenshot shows a terminal window titled "CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)". The terminal displays the following command history:

```
root@localhost ~]# ls
anaconda-ks.cfg test_dict2
root@localhost ~]# grep -r "file"
> ^C
root@localhost ~]#
```

A red arrow points from a red-bordered box containing the text "'↑'키를 누르면..." to the prompt line "root@localhost ~]#".

A-15. 기타 기능

✓ '↑'키를 누른 결과



The screenshot shows a terminal window titled "CentOS 7 64-bit - VMware Workstation 15 Player (Non-commercial use only)". The terminal displays the following commands and output:

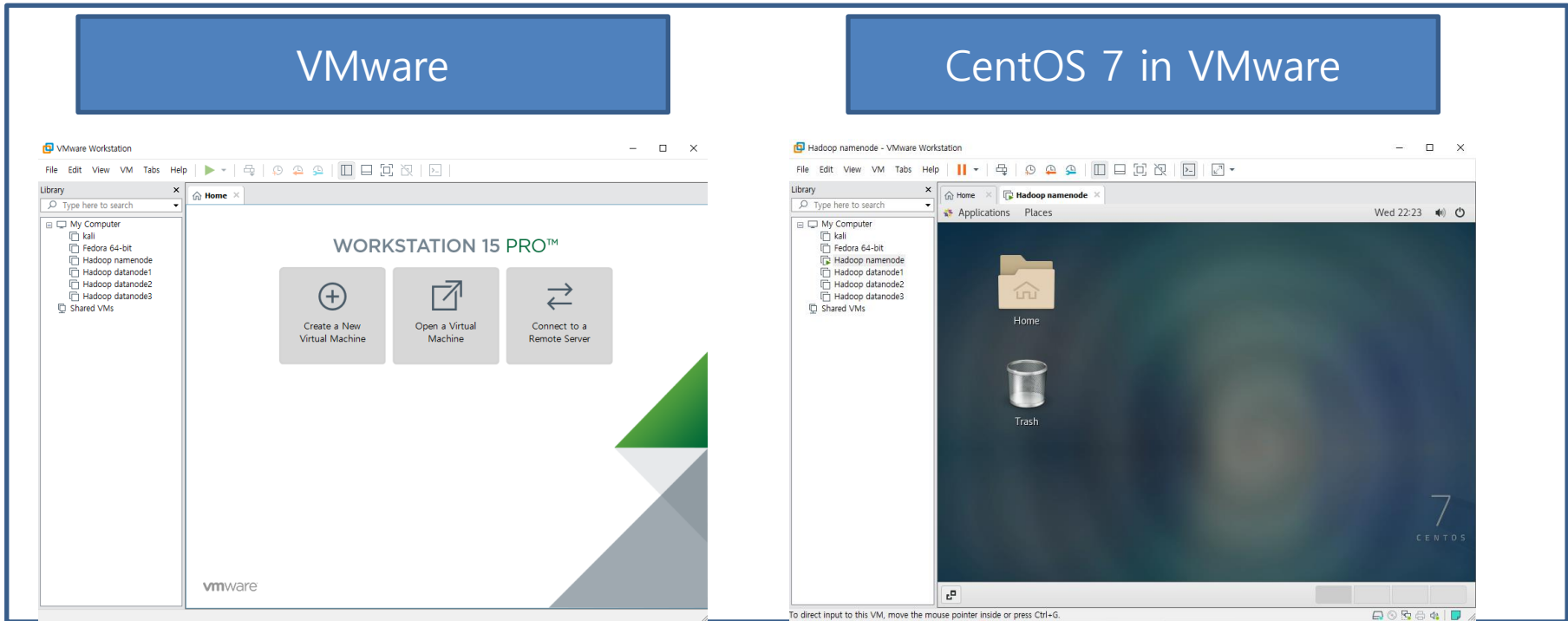
```
[root@localhost ~]# ls
anaconda-ks.cfg  test_dict2
[root@localhost ~]# grep -r "file
> ^C
[root@localhost ~]# grep -r "file
```

A red arrow points from a text box below to the underlined command `grep -r "file`.

이런 식으로 가장 최근에 입력했던
명령어가 자동완성 됨

부록 B. Install Hadoop with VMware

- ✓ VMware를 이용하여 Linux CentOS 7에 Hadoop을 설치하는 방법에 대해 알아보겠습니다.
- ✓ 이 과정은 VMware가 설치되어 있고, VMware 내부에 Linux CentOS 7이 1대 설치되어 있음을 가정합니다.



B-0. 설치 과정 최종 목표

- ✓ 1대의 NameNode와 3대의 DataNode를 구성하여 Hadoop을 실행하고 제대로 동작하는지 확인합니다.
- ✓ 실제로 HDFS, MapReduce를 이용한 WordCount 예제를 실행시킵니다.

Namenode information - Mozilla Firefox

namenode:50070/dfshealth.html#tab-overview

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Overview 'namenode:9000' (active)

Started:	Wed May 06 10:25:53 +0900 2020
Version:	3.2.1, rb3cbb467e22ea829b3808f4b7b01d07e0bf3842
Compiled:	Wed Sep 11 00:56:00 +0900 2019 by rohlthsharmaks from branch-3.2.1
Cluster ID:	CID-36a66021-3d48-4238-b712-57577ce90400
Block Pool ID:	BP-184721169-192.168.50.2-1588698818132

Summary

Security is off.
Safemode is off.

1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).

Heap Memory used 46.13 MB of 60.59 MB Heap Memory. Max Heap Memory is 684.44 MB.

Non Heap Memory used 56.76 MB of 58.16 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	80.94 GB
Configured Remote Capacity:	0 B
DFS Used:	12 KB (0%)
Non DFS Used:	955.94 MB
DFS Remaining:	80 GB (98.85%)
Block Pool Used:	12 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%

[KSC@namenode:/usr/local/hadoop] Namenode information - Mozilla Fir...

B-0. 설치 과정 최종 목표

Namenode information - Mozilla Firefox

Namenode information x +

namenode:50070/dfshealth.html#tab-datanode

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Datanode Information

In service Down Decommissioning Decommissioned Decommissioned & dead
 Entering Maintenance In Maintenance In Maintenance & dead

Datanode usage histogram

Disk usage of each DataNode (%)

In operation

Show 25 entries Search:

Node	Http Address	Last contact	Last Block Report	Capacity	Blocks	Block pool used	Version
✓ datanode1:9866 (192.168.50.3:9866)	http://datanode1:9866	0s	9m	26.98 GB	0	4 KB (0%)	3.2.1
✓ datanode2:9866 (192.168.50.4:9866)	http://datanode2:9866	0s	9m	26.98 GB	0	8 KB (0%)	3.2.1
✓ datanode3:9866 (192.168.50.5:9866)	http://datanode3:9866	0s	9m	26.98 GB	0	4 KB (0%)	3.2.1

[KSC@namenode:/usr/local/hadoop] Namenode information - Mozilla Fir...

B-0. 설치 과정 최종 목표

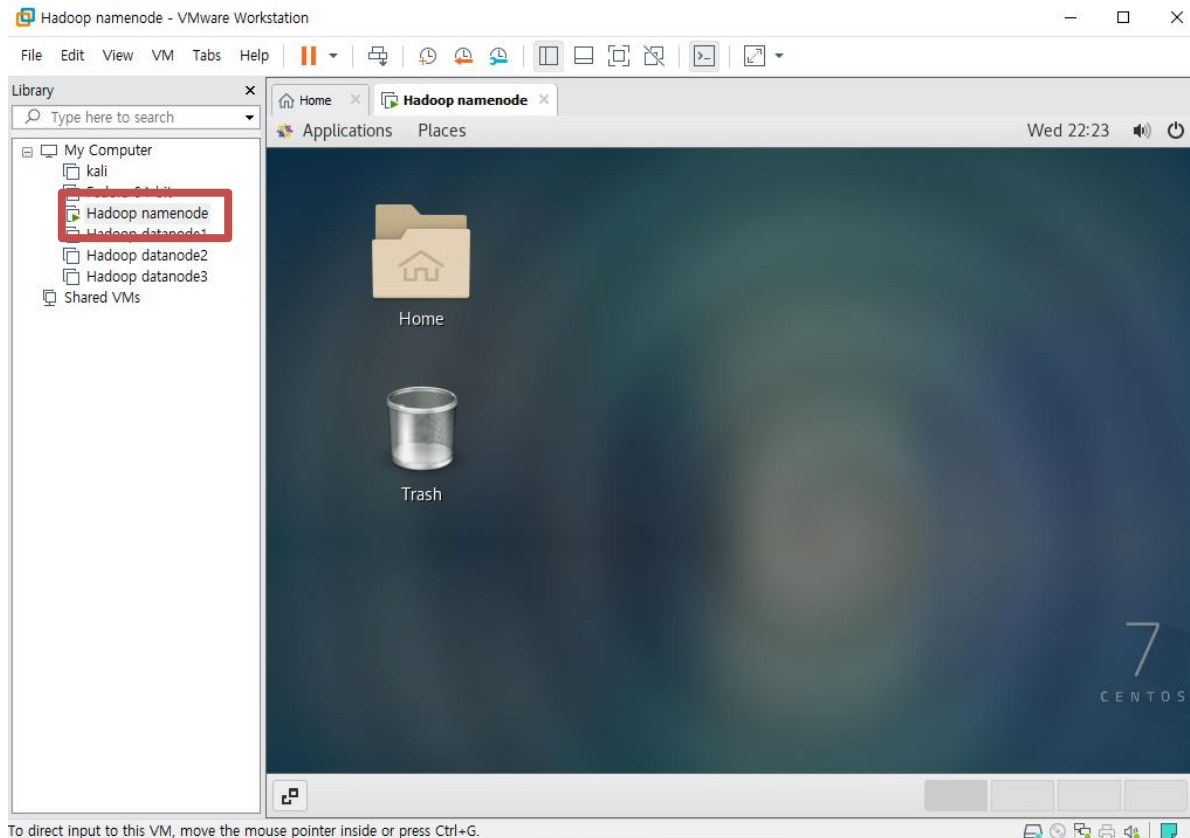
✓ WordCount 예제

```
[root@namenode hadoop]# bin/hdfs dfs -cat /user/hadoop/wordcount/input/file01
2020-05-05 18:12:23,726 INFO sasl.SaslDataTransferClient: SASL encryption trust check:
localHostTrusted = false, remoteHostTrusted = false
Hello World Bye World
[root@namenode hadoop]# bin/hdfs dfs -cat /user/hadoop/wordcount/input/file02
2020-05-05 18:12:29,953 INFO sasl.SaslDataTransferClient: SASL encryption trust check:
localHostTrusted = false, remoteHostTrusted = false
Hello Hadoop Goodbye Hadoop
[root@namenode hadoop]# bin/hdfs dfs -cat /user/hadoop/wordcount/output/part*
2020-05-05 18:10:49,096 INFO sasl.SaslDataTransferClient: SASL encryption trust check:
localHostTrusted = false, remoteHostTrusted = false
Bye 1
Goodbye 1
Hadoop 2
Hello 2
World 2
```

- ✓ "Hello World Bye World"와 "Hello Hadoop Goodbye Hadoop"에서 MapReduce를 활용하여 각 단어의 개수를 출력해봅니다.

B-1. PROLOG

- ✓ VMware에 설치되어 있다고 가정한 CentOS 7의 이름을 `Hadoop namenode`라고 하겠습니다. (이 부분은 다르게 하시더라도 상관 없습니다.)
- ✓ 최대한 중복 작업을 피하기 위해 아래에 있는 datanode1, 2, 3은 필요할 때 만들도록 하겠습니다.
- ✓ 기본적으로 모든 작업은 **root**계정으로 진행합니다.



B-2. Java 설치 및 환경 변수 설정

- ✓ 먼저, Hadoop은 java를 기반으로 제작되었기 때문에 Hadoop 명령을 이용할 때 java가 동작할 수 있도록 java를 설치합니다. (호환성을 위해 jdk version은 8로 맞춰 주시기 바랍니다.)

```
# rpm -qa | grep java
```

```
[root@localhost hadoop]# rpm -qa | grep java
tzdata-java-2018e-3.el7.noarch
java-1.8.0-openjdk-1.8.0.181-7.b13.el7.x86_64
javassist-3.16.1-10.el7.noarch
javapackages-tools-3.4.1-11.el7.noarch
java-1.7.0-openjdk-headless-1.7.0.191-2.6.15.5.el7.x86_64
javamail-1.4.6-8.el7.noarch
python-javapackages-3.4.1-11.el7.noarch
java-1.7.0-openjdk-1.7.0.191-2.6.15.5.el7.x86_64
java-1.8.0-openjdk-headless-1.8.0.181-7.b13.el7.x86_64
```

`rpm -qa | grep java` 명령은 현재 CentOS에 설치되어 있는 java 관련 패키지를 보여줍니다. 본 설치과정에서는 새롭게 jdk를 받아 환경 변수를 설정할 것이기 때문에 관련 패키지들을 모두 제거하겠습니다.

B-2. Java 설치 및 환경 변수 설정

`yum`은 CentOS의 패키지 관리자인데 이것을 이용하여 java관련 패키지를 제거합니다.

```
# yum remove tzdata-java.noarch  
# yum remove javapackages-tools.noarch  
# yum remove python-javapackages.noarch
```

```
# rpm -qa | grep java
```

이제 위의 명령어를 입력하면 아무것도 나오지 않을 것입니다.

```
[root@localhost hadoop]# rpm -qa | grep java  
[root@localhost hadoop]#
```

이후 Linux의 기본 브라우저인 FireFox로 Oracle에 접속한 후, JDK 8을 tar.gz로 내려받습니다. (이때, oracle 계정이 필요합니다.)

B-2. Java 설치 및 환경 변수 설정







<https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>

위의 url로 접속하시면 아래와 같이 jdk 다운로드 페이지로 이동하실 수 있습니다.
(주의 : url은 시간이 지남에 따라 항상 변동될 수 있습니다.)

Linux x64 Compressed Archive (186.09MB)를 다운로드합니다.

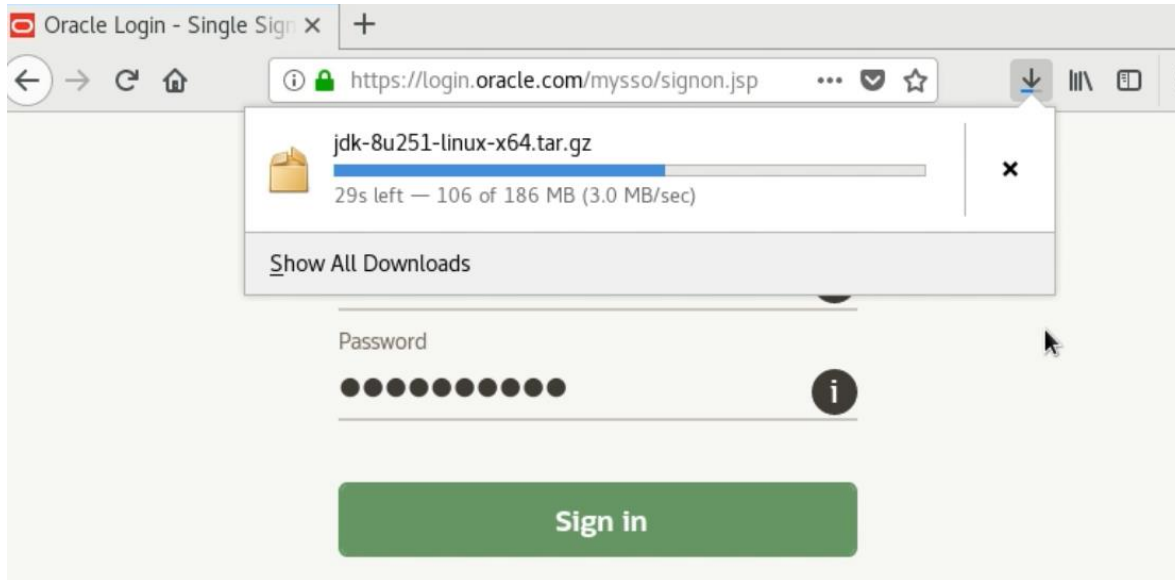
Java SE Development Kit 8u251

This software is licensed under the [Oracle Technology Network License Agreement for Oracle Java SE](#)

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.87 MB	 jdk-8u251-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	69.77 MB	 jdk-8u251-linux-arm64-vfp-hflt.tar.gz
Linux x86 RPM Package	171.71 MB	 jdk-8u251-linux-i586.rpm
Linux x86 Compressed Archive	186.6 MB	 jdk-8u251-linux-i586.tar.gz
Linux x64 RPM Package	171.16 MB	 jdk-8u251-linux-x64.rpm
Linux x64 Compressed Archive	186.09 MB	 jdk-8u251-linux-x64.tar.gz

B-2. Java 설치 및 환경 변수 설정

다운로드를 위해서는 Cookie를 허용하고, Oracle 계정으로 로그인 해 주셔야 합니다.
(Oracle 계정 생성 및 JDK 다운로드는 무료입니다.)



이렇게 다운로드를 해주시고 나면 다운받은 jdk-8u251-linux-x64.tar.gz가 있는 디렉터리로 이동합니다. (일반적으로 /Home/[user]/Download/ 에 위치합니다.)
[Download] # ls 를 통해 파일이 존재하는 것을 확인합니다.

```
[root@namenode 다운로드]# ls
jdk-8u251-linux-x64.tar.gz
```

B-2. Java 설치 및 환경 변수 설정

```
# tar zxvf jdk-8u251-linux-x64.tar.gz
```

`jdk-8u251-linux-x64.tar.gz`가 존재하는 위치에서 위 명령을 입력하여 압축을 해제합니다.

```
[root@namenode 다운로드]# ls
jdk-8u251-linux-x64.tar.gz  jdk1.8.0_251
[root@namenode 다운로드]# mv jdk1.8.0_251/ /usr/local/jdk
```

압축을 해제하면 `jdk1.8.0_251`이라는 디렉터리가 생성된 것을 확인할 수 있습니다.

```
# mv jdk1.8.0_251/ /usr/local/jdk
```

`jdk1.8.0_251`을 /usr/local/jdk로 이동시킵니다.

```
# vi /etc/profile
```

/etc/profile을 vi 편집기로 편집합니다.

```
export JAVA_HOME=/usr/local/jdk
HADOOP_HOME=/usr/local/hadoop
PATH=$PATH: $JAVA_HOME/bin: $HADOOP_HOME/bin
```

/etc/profile의 최하단에 위에 보이시는 3행을 입력합니다.
(vi 편집기 사용법까지는 기재하지 않겠습니다.)

B-2. Java 설치 및 환경 변수 설정

```
# source /etc/profile
```

방금 변경한 사항을 시스템에 적용합니다.

```
# java -version
```

```
[root@namenode local]# source /etc/profile
[root@namenode local]# java -version
java version "1.8.0_251"
Java(TM) SE Runtime Environment (build 1.8.0_251-b08)
Java HotSpot(TM) 64-Bit Server VM (build 25.251-b08, mixed mode)
```

위와 같이 버전이 출력되면 `B-2. Java 설치 및 환경 변수 설정`은 완료입니다.

B-3. 방화벽 설정

- ✓ Linux를 사용할 때는 보안을 위해 항상 방화벽을 잘 설정해야 합니다.
- ✓ 먼저, 사용하는 IP 대역을 192.168.50.0/24 (subnetmask=255.255.255.0)으로 가정합니다.
- ✓ 간단하게 http(80번 포트)를 모든 IP에 대해 허용하고, ssh(22번 포트)를 192.168.50.0/24 대역에 한해서 허용하도록 하겠습니다.

```
# firewall-cmd --permanent --add-port=80/tcp
```

위 명령으로 80번 포트를 개방합니다.

```
# firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source  
address=192.168.50.0/24 port port="22" protocol="tcp" accept'
```

위 명령으로 192.168.50.0/24 대역에 해당하는 IP에 한해서 SSH 접근을 허용합니다.

```
# firewall-cmd --reload
```

변경한 설정이 적용될 수 있도록 방화벽을 reload 해줍니다.

```
# firewall-cmd --list-all
```

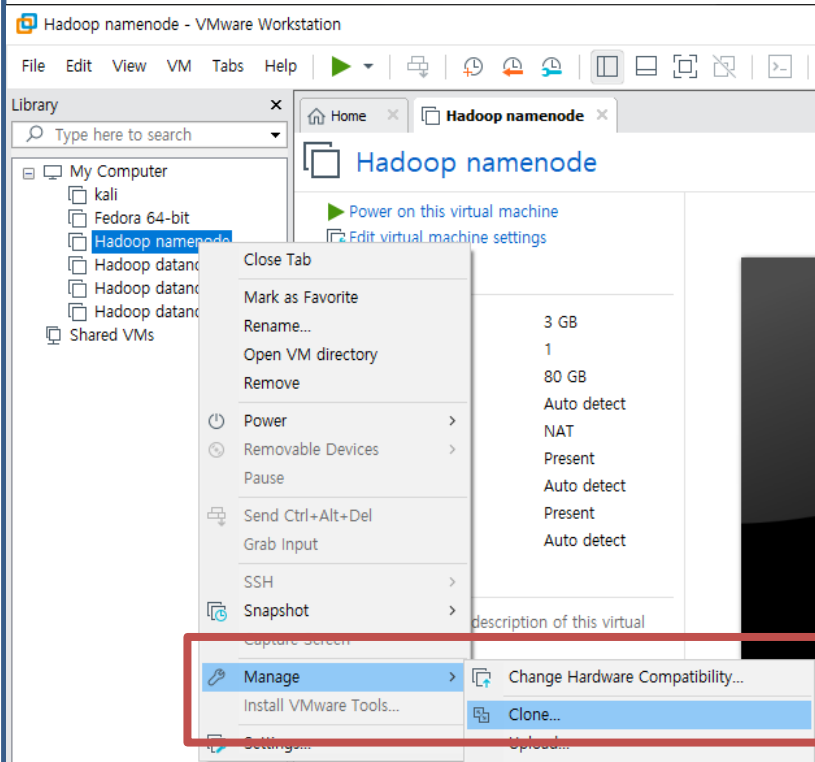
변경한 설정이 반영되었는지 확인합니다.

```
# systemctl restart firewalld      // 방화벽을 실행합니다.  
# systemctl enable firewalld       // 재부팅해도 방화벽이 실행되도록 설정합니다.
```

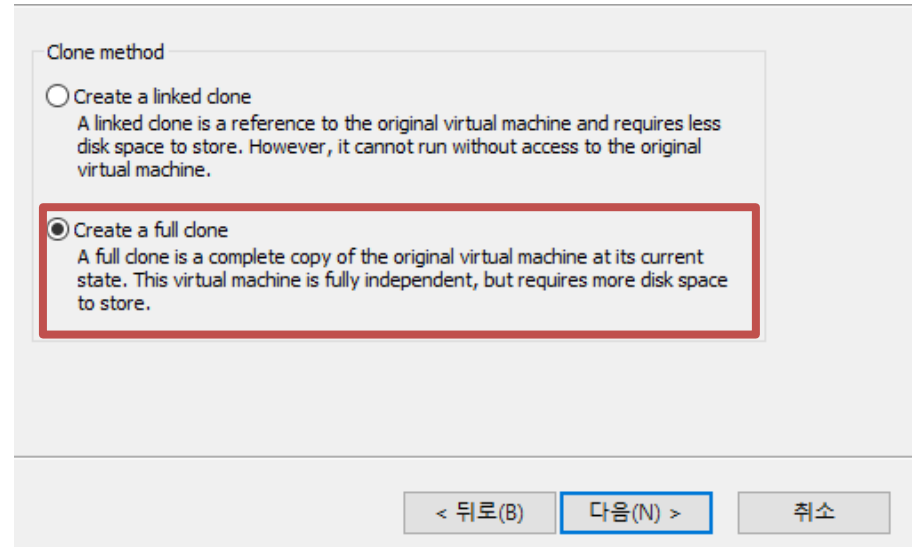
B-4. Clone Namenode & Create Datanode1

- ✓ 이번에는 Hadoop에서 DataNode로 사용될 VM(가상머신)을 생성하겠습니다.
- ✓ 이 단계에서는 일단 1대만 생성합니다.

Namenode를 우클릭한 후, Manage > Clone을 클릭합니다.



중간에 Clone method를 선택하는 옵션이 나오는데, full clone을 선택합니다.



마지막으로, 이름을 적당히 설정한 후, 복제를 진행합니다. (예: Hadoop datanode1)

B-4. Clone Namenode & Create Datanode1

아래와 같이 VM이 생성되었다면 B-4 단계 완료입니다.

Home x Hadoop namenode x Hadoop datanode1 x

Hadoop datanode1

▶ Power on this virtual machine
 ▶ Edit virtual machine settings

▼ Devices

Memory	2 GB
Processors	1
Hard Disk (SCSI)	80 GB
CD/DVD (IDE)	Auto detect
Network Adapter	NAT
USB Controller	Present
Sound Card	Auto detect
Printer	Present
Display	Auto detect

▼ Description

Type here to enter a description of this virtual machine.

▼ Virtual Machine Details

State: Powered off
Snapshot: jdk, hadoop deploy for datanode1
Configuration file: C:\Users\WKSC\Desktop\HadoopVMs\Hadoop DataNode1\Hadoop datanode1.vmx
Hardware compatibility: Workstation 15.x virtual machine
Primary IP address: Network information is not available

B-5. Network Configuration between Nodes

- ✓ 여기서는 Namenode와 Datanode1의 네트워크를 설정하고, ssh를 이용한 통신이 가능하도록 설정하겠습니다.
- ✓ 본 과정에서는 VMware의 가상 네트워크를 조금 조작하여 192.168.50.0/24 대역을 사용하도록 설정하고, Namenode의 IP를 192.168.50.2로, Datanode1의 IP를 192.168.50.3으로 조정했습니다.
- ✓ 각자의 VMware의 설정과 다를 수 있기 때문에 `ifconfig`로 자신에게 맞는 IP를 사용해주시기 바랍니다.

```
# ifconfig
```

```
[root@localhost .ssh]# ifconfig
ens33: flags=4163<UP, BROADCAST, RUNNING, MULTICAST> mtu 1500
    inet 192.168.50.2 netmask 255.255.255.0 broadcast 192.168.50.255
    inet6 fe80::e446:6860:5db1:2b66 prefixlen 64 scopeid 0x20<link>
    inet6 fe80::9877:daa1:6702:9fd prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:6b:2f:d1 txqueuelen 1000 (Ethernet)
    RX packets 321 bytes 27891 (27.2 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 53 bytes 7307 (7.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Namenode의 IP가 `192.168.50.2`인 것을 확인할 수 있습니다.
동일한 방식으로 복제해둔 Datanode1의 IP도 확인합니다.
(필자의 경우, Datanode1의 IP가 `192.168.50.3`입니다.)

B-5. Network Configuration between Nodes

- ✓ B-5 단계부터는 Namenode와 Datanode1을 함께 사용하므로 어떤 머신에서 작업하는 것인지 확실히 구분하시고 진행해야 합니다.
- ✓ 지금부터 설정하는 것은 나중에 복제할 Datanode2, 3을 위해 그것의 설정까지 함께 해 두도록 하겠습니다. 존재하지 않는 datanode2, 3이 설정에 등장해도 당황하지 않고 진행하시면 됩니다.

```
# vi /etc/hosts
```

```
[root@namenode KSC]# cat /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.50.2 namenode
192.168.50.3 snamenode
192.168.50.3 datanode1
192.168.50.4 datanode2
192.168.50.5 datanode3
```

`/etc/hosts` 파일의 하단에 위 5행을 추가해줍니다. (snamenode는 추가하지 않으셔도 됩니다.)

<Namenode, Datanode1 모두 설정해 주셔야 합니다.>

B-5. Network Configuration between Nodes

```
# vi /etc/hostname
```

```
[root@namenode KSC]# cat /etc/hostname  
namenode  
[root@namenode KSC]# hostname  
namenode
```

`/etc/hostname` 파일의 내용을 namenode로 바꿔줍니다.

<Datanode1에서는 `/etc/hostname` 파일의 내용을 datanode1으로 바꿔줍니다.>

```
# hostname
```

위 명령을 입력했을 때 각자 설정해 준 이름으로 나오면 완료입니다.
(설정 직후에는 적용되지 않았을 수 있습니다. **재부팅** 하시면 됩니다.)

```
# ping datanode1
```

Namenode에서 위 명령을 입력해 datanode1과 통신을 확인합니다.

```
[root@namenode KSC]# ping datanode1  
PING datanode1 (192.168.50.3) 56(84) bytes of data.  
64 bytes from snamenode (192.168.50.3): icmp_seq=1 ttl=64 time=0.335 ms  
64 bytes from snamenode (192.168.50.3): icmp_seq=2 ttl=64 time=0.209 ms  
64 bytes from snamenode (192.168.50.3): icmp_seq=3 ttl=64 time=0.377 ms  
64 bytes from snamenode (192.168.50.3): icmp_seq=4 ttl=64 time=0.361 ms  
64 bytes from snamenode (192.168.50.3): icmp_seq=5 ttl=64 time=0.320 ms  
^C  
--- datanode1 ping statistics ---  
5 packets transmitted, 5 received, 0% packet loss, time 4000ms  
rtt min/avg/max/mdev = 0.209/0.320/0.377/0.061 ms
```

중간에 ctrl+c 로 끊어주셔야 합니다.

loss가 0이면 정상입니다.

B-5. Network Configuration between Nodes

- ✓ 이제는 SSH 연결을 진행해보도록 하겠습니다.
- ✓ 노드 간에 SSH 연결이 되어 있어야 Hadoop의 분산 시스템이 정상적으로 동작합니다.

DataNode1

```
# ssh-keygen -t rsa
```

ssh key를 생성합니다.

```
[root@localhost ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

입력창이 나오는데 당황하지 않고 enter를 3번 눌러줍니다.

```
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256: 0YQYiMq7hTgPP7cINSuFEr5qPZSjV/ezbXD1P2doRdM root@localhost.localdomain
The key's randomart image is:
+--[RSA 2048]-----+
|  . . . o .       |
|+ . . . . .       |
|o+ . . . . .E|
|+. * . . S . . . |
|== B . . o . . . |
|o*B o . . o . o .|
|. *+=. . o . . o .+|
|o ooo. . . + . . |
+---[SHA256]-----+
```

좌측 그림과 같이 출력되면 정상적으로 키가 생성된 것입니다.

B-5. Network Configuration between Nodes

```
# ls ~/.ssh
```

위 명령을 입력했을 때 `id_rsa`와 `id_rsa.pub`이 존재하는 것을 확인합니다.

NameNode

```
# ssh-keygen -t rsa
```

NameNode에서도 동일하게 ssh key를 생성합니다.

```
# cd ~/.ssh
```

NameNode에서는 작업할 것이 있기 때문에 `~/.ssh`로 이동합니다.

```
[root@localhost .ssh]# ls
id_rsa id_rsa.pub
[root@localhost .ssh]# cp id_rsa.pub authorized_keys
[root@localhost .ssh]# ls
authorized_keys id_rsa id_rsa.pub
[root@localhost .ssh]# ssh root@datanode1 cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

ls로 `id_rsa`와 `id_rsa.pub`이 생성되었는지 확인하고, `id_rsa.pub`을 `authorized_keys`로 복사합니다. 그리고 다음 명령으로 datanode1에 있는 `id_rsa.pub`을 namenode의 `authorized_keys`에 추가합니다.

B-5. Network Configuration between Nodes

```
[root@localhost .ssh]# ssh root@datanode1 cat ~/.ssh/id_rsa.pub W
> >> ~/.ssh/authorized_keys
The authenticity of host 'datanode1 (192.168.50.3)' can't be established.
ECDSA key fingerprint is SHA256:WJjcIb22p+db4nHPixnLSZgJ5H3//Je3iE6pf3p3DYU.
ECDSA key fingerprint is MD5: 82: ac: 7f: 3c: 68: ff: e2: d5: 94: 70: 5f: 44: 33: 9b: d5: c0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'datanode1,192.168.50.3' (ECDSA) to the list of known hosts.
root@datanode1's password:
```

위와 같이 질의문이 나오는데, `yes` 해주시고 datanode1의 root password를 입력해줍니다.

```
# cat authorized_keys
```

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQADK3lBsfcWU8nyQQ6D7/dNWnr6L32vmng86w8efXvMTwmYN+Gi
CR1YXCwddH8WmjjiY52sBk01Uh8JYpcYy2Iz4VAecBjvs82VKjivwGe4fk6hfXoodYwmgImHKvh371+dB20/3x
RzEbkj2DTmV5hLpn5LTt0w6L6SNgiDRl306PoGNeEE7yvaZSQ+oZ+6bksNkv1VOMPcNwimn5ABGQVrMyYAg4qNZ
aMW6vMydstGKzGrKQk+KMoPk4VPw6XxfzwU/mK8+CzbMeFTn9oDFrQB+pJ8MUu1eCI+Jv+6x0DPw/DmEXK/aQ3L
4A7zBemMQ5ULFqhgF0CVBf7r8eFU4ppL root@namenode
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQADK3lBsfcWU8nyQQ6D7/dNWnr6L32vmng86w8efXvMTwmYN+Gi
vMmJzuhNjJCZYLzea8r0P5+ehwf6nJf4DxUfofw5L4yhX3AK0WjzK6quMgh9EAsxakrxVjxvRhBJn3J5/dqhdsm
js0Z7zco4kWgvdUj5YCRNeTtBhCoruNZS0Gp61mzK62AzdEbLx7B450i/bBsiB0/cCJ75WR1VA8HKR2e1XrLdb9
r70QwSoorXuUrWqMMLL0k+yFleYAKnK073Tv/jC+enBqTnHtcHpQTZ2QuxBAwwb+8YNGCJl10ydbLwKmVPX5ugc
4ox2A9wl5LgooH2ra6eejJt202SmfYIn root@datanode1
```

이후 `authorized_keys`의 내용을 보시면 위와 같이 `root@datanode1`에 대한 내용이 존재하는 것을 보실 수 있습니다.



B-5. Network Configuration between Nodes

```
# scp -rp authorized_keys root@datanode1:~/.ssh/authorized_keys
```

```
[root@localhost . ssh]# scp -rp authorized_keys root@datanode1: ~/.ssh/authorized_keys  
root@datanode1's password:  
authorized_keys                                100% 1187      1.3MB/s   00: 00
```

위 명령으로 authorized_keys를 datanode1에 복사해줍니다.

```
# ssh datanode1 date
```

```
[root@localhost . ssh]# ssh datanode1 date  
2020. 05. 05. (화) 14:26:03 KST
```

다음과 같이 ssh를 통해 datanode1에 접근해서 현재 날짜를 물어봤을 때 **datanode1**의 **password**를 **요구하지 않고** 날짜를 출력해주면 SSH 설정 완료입니다.

B-6. Hadoop 설치

- ✓ 이번에는 Hadoop을 설치하고 설정 파일을 환경과 용도에 맞게 설정하도록 하겠습니다.
- ✓ Hadoop 다운로드 및 설정 파일 변경을 **Namenode에서** 진행한 후, Datanode1으로 복사하겠습니다.

```
# wget "http://apache.mirror.cdnetworks.com/hadoop/common/stable/Hadoop-3.2.1.tar.gz"
```

wget명령을 이용하여 해당 미러사이트에서 Hadoop package를 다운로드합니다.
(미러사이트의 주소, 저장된 Hadoop의 버전은 언제든지 변경될 수 있습니다.)

본 설치과정에서는 2020-05-07 기준 최신 stable 버전인 3.2.1 버전을 이용합니다.

```
[root@localhost ~]# wget "http://apache.mirror.cdnetworks.com/hadoop/common/stable/hadoop-3.2.1.tar.gz"
--2020-05-05 14:36:54-- http://apache.mirror.cdnetworks.com/hadoop/common/stable/hadoop-3.2.1.tar.gz
Resolving apache.mirror.cdnetworks.com (apache.mirror.cdnetworks.com)... 14.0.101.165
Connecting to apache.mirror.cdnetworks.com (apache.mirror.cdnetworks.com)|14.0.101.165|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 359196911 (343M) [application/x-gzip]
Saving to: 'hadoop-3.2.1.tar.gz'

100%=====>] 359,196,911 9.19MB/s in 31s
```

```
[root@localhost ~]# ls
anaconda-ks.cfg  initial-setup-ks.cfg  다운로드  바탕화면  사진  음악
hadoop-3.2.1.tar.gz  공개  문서  비디오  서식
```

ls를 통해 파일이 받아진 것을 확인합니다.

B-6. Hadoop 설치

```
# tar xvf hadoop-3.2.1.tar.gz
```

`tar xvf` 명령을 통해 압축을 해제합니다.

```
[root@localhost ~]# ls
anaconda-ks.cfg  hadoop-3.2.1.tar.gz  공개  문서  비디오  서식
hadoop-3.2.1     initial-setup-ks.cfg  다운로드  바탕화면  사진  음악
```

압축이 해제되어 `hadoop-3.2.1`이라는 디렉터리가 생성된 것을 볼 수 있습니다.

```
# mv hadoop-3.2.1 /usr/local/hadoop
```

`hadoop-3.2.1`을 `/usr/local/hadoop`으로 이동시킵니다.

```
[root@localhost ~]# mkdir -p /home/hadoop/hdfs/data
[root@localhost ~]# mkdir -p /home/hadoop/hdfs/temp
[root@localhost ~]# mkdir -p /home/hadoop/hdfs/name
```

`mkdir` 명령을 통해 해당 디렉터리 3개를 생성합니다. 이 디렉터리들은 Hadoop의 분산 파일관리 시스템의 그릇에 해당합니다.

(-p 옵션을 사용하면 중간에 존재하지 않는 `hadoop`, `hdfs` 디렉터리가 자동으로 생성된 후, `data`, `temp`, `name` 디렉터를 생성합니다.)

B-6. Hadoop 설치

```
# cd /usr/local/hadoop/etc/hadoop
```

```
# ls
```

20개가 넘는 설정파일들이 존재합니다. 여기서 본 과정에서는 5개의 파일을 수정합니다. [`hadoop-env.sh`, `core-site.xml`, `hdfs-site.xml`, `mapred-site.xml`, `workers`]

```
# vi hadoop-env.sh
```

```
# For example, to limit who can execute the namenode command,  
# export HDFS_NAMENODE_USER=hdfs
```

```
export HDFS_NAMENODE_USER=root  
export HDFS_DATANODE_USER=root  
export HDFS_SECONDARYNAMENODE_USER=root  
export YARN_RESOURCEMANAGER_USER=root  
export YARN_NODEMANAGER_USER=root
```

```
export JAVA_HOME=/usr/local/jdk  
export HADOOP_HOME_WARN_SUPPRESS=1
```

해당 파일의 최하단에 위의 7행을 추가하고, 저장합니다.

B-6. Hadoop 설치

vi core-site.xml

```
<!-- Put site-specific property overrides in this file. -->
```

```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://namenode:9000</value>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/home/hadoop/hdfs/temp</value>
  </property>
</configuration>
```

해당 파일의 <configuration> 블록 내부를 좌측과 같이 작성합니다.

vi hdfs-site.xml

```
<!-- Put site-specific property overrides in this file. -->
```

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>2</value>
  </property>
  <property>
    <name>dfs.name.dir</name>
    <value>/home/hadoop/hdfs/name</value>
  </property>
  <property>
    <name>dfs.data.dir</name>
    <value>/home/hadoop/hdfs/data</value>
  </property>
  <property>
    <name>dfs.http.address</name>
    <value>namenode:50070</value>
  </property>
</configuration>
```

해당 파일의 <configuration> 블록 내부를 좌측과 같이 작성합니다.

(첫번째 property의 value가 2가 되어있는데, 완전 분산 모드로 구성하기 위해서는 3으로 설정합니다.)

B-6. Hadoop 설치

```
# vi mapred-site.xml
```

```
<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>namenode:9001</value>
  </property>
  <property>
    <name>mapred.local.dir</name>
    <value>/home/hadoop/hdfs/mapred</value>
  </property>
  <property>
    <name>mapred.system.dir</name>
    <value>/home/hadoop/hdfs/mapred</value>
  </property>
</configuration>
```

해당 파일의
<configuration> 블록 내부
를 좌측과 같이 작성합니다.

```
# vi workers
```

```
datanode1
datanode2
datanode3
```

workers 파일은 기본적으로 비어 있을 것입니다.
좌측의 3행을 추가합니다.

이제 모든 설정이 완료되었습니다.

B-6. Hadoop 설치

```
# cd /usr/local  
# tar zcvf hadoop.tar.gz ./hadoop
```

/usr/local로 이동한 후, 설정을 변경한 hadoop 디렉터리를 `hadoop.tar.gz`로 압축합니다.

```
# scp -rp hadoop.tar.gz datanode1:/usr/local/hadoop.tar.gz
```

압축한 `hadoop.tar.gz` 파일을 datanode1의 /usr/local/로 복사합니다.

DataNode1

```
# cd /usr/local  
# tar zxvf hadoop.tar.gz
```

Datanode1에서 /usr/local로 이동하시면 Namenode에서 보낸 압축파일이 있습니다.
`tar zxvf`를 통해 압축을 해제합니다.

여기까지 모든 과정이 진행되었다면 Hadoop을 실행하기 전에 Datanode1을 Clone해서 Datanode2, 3을 만들겠습니다.

B-7. Clone Datanode1 & Create DataNode2, 3

- ✓ 이번에는 Datanode1을 Clone해서 Datanode2, 3을 만들겠습니다.
- ✓ `B-4` 과정과 동일하게 진행하되, **Datanode1을 Clone**하는 것입니다. (Full clone)
- ✓ 생성된 Datanode2, 3에서 각자 조금씩 수정해 줄 부분이 있습니다. (hostname, ssh 설정)
- ✓ 각자의 **Datanode2, 3의 IP를 잘 확인**하셔야 합니다. 본 과정에서는 **Datanode2, 3의 IP는 각각 `192.168.50.4`, `192.168.50.5`**입니다. (ifconfig로 확인)

```
# vi /etc/hostname
```

Datanode2에서는 datanode2로, Datanode3에서는 datanode3으로 지정합니다.
(설정 후 재부팅 해 주셔야 합니다.)

```
[root@datanode2 ~]# hostname  
datanode2
```

`hostname` 명령으로 확인했을 때 위와 같이 출력되면 정상입니다.

```
# cd  
# ssh-keygen -t rsa
```

Datanode2, 3에서 각각 위 명령을 실행하여 SSH key를 생성합니다.



B-7. Clone Datanode1 & Create DataNode2, 3

Namenode

```
# cd ~/.ssh
# ssh root@datanode2 cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
# ssh root@datanode3 cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
# cat authorized_keys
```

```
RzEbkj2DTmV5hLpn5LTt0w6L6SngiDRl306PoGNeEE7yvaZSQ+oZ+6bksNkv1V0MPcNwimn5ABGQVrMyYAg4qNZ
aMW6vMydstGKzGrKQk+KMoPk4VPw6XxfzwU/mK8+CzhMeFTn9oDFrQB+pJ8MUuleCI+Jv+6x0DPw/DmEXK/aQ3L
4A7zBemMQ5ULFqhgF0CVBf7r8eFU4ppL root@namenode
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQAClXhU3JRX09BwA3LTdKT8yVmenToCrrKMMGivDs0FC9bspuV2
vMmJzuhNjJCZYlzea8r0P5+ehwf6nJf4DxUfofw5L4yhX3AK0WjzK6quMgh9EAsxakrxVjxvRhBJn3J5/dqhdsM
js0Z7zco4kWgvdUj5YCRNeTtBhCoruNZS0Gp6lmzK62AzdEbLx7B450i/bBsiB0/cCJ75WR1VA8HKR2e1XrLdb9
r70QwSoorXuUrWqMMLL0k+yFleYAKnK073Tv/jC+enBqTnHtcHpQTZ2QuxBAwwb+8YNGCJl10ydBlwKmVPX5ugo
4ox2A9wl5Lg0oH2ra6eejJt202SmfyIn root@snamenode
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQAClXhU3JRX09BwA3LTdKT8yVmenToCrrKMMGivDs0FC9bspuV2
vMmJzuhNjJCZYlzea8r0P5+ehwf6nJf4DxUfofw5L4yhX3AK0WjzK6quMgh9EAsxakrxVjxvRhBJn3J5/dqhdsM
js0Z7zco4kWgvdUj5YCRNeTtBhCoruNZS0Gp6lmzK62AzdEbLx7B450i/bBsiB0/cCJ75WR1VA8HKR2e1XrLdb9
r70QwSoorXuUrWqMMLL0k+yFleYAKnK073Tv/jC+enBqTnHtcHpQTZ2QuxBAwwb+8YNGCJl10ydBlwKmVPX5ugo
4ox2A9wl5Lg0oH2ra6eejJt202SmfyIn root@datanode1
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQAClQzn1tStKawLUI95MahQ9XAWkeVlMk8M8X91P1kcodl+sOWk
Qau6eKbZB54ypRWJod2jGgYgrQrYmVWLlKJcWrlYcZ2SLwzSDDyqKrfw/18HW7CK/yTiYiPZ9swxHRCuRpA7Sd
sCCWk021GFKFgtZmBtXTIfomQMSIjsuQCjjKoL4Hb+EkRiCMD4K8zTkuR/U3oXNZu7fKmkuy/MPZkAipw2XW3
44jX+B52Qq49eKl3HF7alje1yf3fjjEyMBQswaRim08vbi1VZVs+10MH1v7iaEgz0GZgp05TLf6S81yDJlfuo0S
jfC03G3bsZKJATJJudMDNB9FTH1aGuGB root@datanode2
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQACuamNoJ08N6NKOYQEuiub5Vjzumn566u0NdxLDo5mKL6e45Nn
CguHsQM6+VBNTGPKY5gzI+WLMe1ACXImoaD/7Nn5yfb0s/UHfbbNdbR1/Yd60r2piuBPbNfNcgUr8Xt7HR6T/4h
ZMRqHyNmMhZs593cqMIXuXOK/ryNsUdVSwUp6tkjdf0+rDFRGRZvtKueyuTJNQ2CJn9m1j43kj2qIY03vtHB6S
V3wcGW00HvQQZP/jo1ofh/oIZl03YzvED12DIaEb/N9m6e9KKNGhoE2HQYH0ch9CgEZjJm74ezbNAZQmQqfivs
mW+lzwBEDNHhlt73/uBJXlg61VfLrUDH root@datanode3
```

`authorized_keys`를 확인했을 때 위와 같이 되어 있으면 정상입니다.
(root@snamenode는 없어도 됩니다.)



B-7. Clone Datanode1 & Create DataNode2, 3

Namenode

```
# scp -rp authorized_keys root@datanode1:~/.ssh/authorized_keys  
# scp -rp authorized_keys root@datanode2:~/.ssh/authorized_keys  
# scp -rp authorized_keys root@datanode3:~/.ssh/authorized_keys
```

`authorized_keys`를 Datanode들에 각각 복사합니다.

```
# ssh datanode1 date  
# ssh datanode2 date  
# ssh datanode3 date
```

Namenode에서 ssh를 통해 datanodes에 접근해봅니다. 이때, 패스워드를 질의하지 않고 시스템 날짜를 보여주면 B-7 단계까지 완료입니다.

B-8. Hadoop 실행

Namenode

```
# cd /usr/local/hadoop  
# ./bin/hadoop namenode -format
```

위 명령으로 Hadoop Namenode를 초기화합니다.

```
# ./sbin/start-all.sh
```

Hadoop의 HDFS와 MapReduce 서비스를 실행합니다.

각 노드에서

```
# jps
```

간단하게 각 노드에서 jps 명령을 통해 Hadoop이 정상 실행되고 있는지 확인하실 수 있습니다.

Namenode의 경우, Namenode가 출력되어야 하고,
Datanode들의 경우, Datanode가 출력되어야 합니다.

B-8. Hadoop 실행

Namenode

Firefox에서 "namenode:50070/"으로 접속하면 아래와 같이 Hadoop이 실행중임을 확인하실 수 있습니다.

Overview 'namenode:9000' (active)

Started:	Wed May 06 10:25:53 +0900 2020
Version:	3.2.1, rb3cbbb467e22ea829b3808f4b7b01d07e0bf3842
Compiled:	Wed Sep 11 00:56:00 +0900 2019 by rohithsharmaks from branch-3.2.1
Cluster ID:	CID-36a66021-3d48-4238-b712-57577ce90400
Block Pool ID:	BP-184721169-192.168.50.2-1588698818132

Summary

Security is off.
Safemode is off.
1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).
Heap Memory used 46.13 MB of 60.59 MB Heap Memory. Max Heap Memory is 684.44 MB.
Non Heap Memory used 56.76 MB of 58.16 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	80.94 GB
Configured Remote Capacity:	0 B
DFS Used:	12 KB (0%)
Non DFS Used:	955.94 MB
DFS Remaining:	80 GB (98.85%)
Block Pool Used:	12 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%

B-8. Hadoop 실행

Datanodes로 들어가시면 작동중인 datanodes도 확인하실 수 있습니다.

Namenode information - Mozilla Firefox

namenode:50070/dfshealth.html#tab-datanode

Hadoop Overview **Datanodes** Datanode Volume Failures Snapshot Startup Progress Utilities

Datanode Information

✓ In service
⬇ Down
⚙ Decommissioning
⚠ Decommissioned
⬇ Decommissioned & dead
✂ Entering Maintenance
✂ In Maintenance
✂ In Maintenance & dead

Datanode usage histogram

In operation

Show 25 entries Search:

Node	Http Address	Last contact	Last Block Report	Capacity	Blocks	Block pool used	Version
✓ datanode1:9866 (192.168.50.3:9866)	http://datanode1:9864	0s	9m	26.98 GB	0	4 KB (0%)	3.2.1
✓ datanode2:9866 (192.168.50.4:9866)	http://datanode2:9864	0s	9m	26.98 GB	0	8 KB (0%)	3.2.1
✓ datanode3:9866 (192.168.50.5:9866)	http://datanode3:9864	0s	9m	26.98 GB	0	4 KB (0%)	3.2.1

[KSC@namenode:usr/local/hadoop] Namenode information - Mozilla Fir...

B-9. WordCount 예제 실행

- ✓ `B-8` 단계까지 수행하시면 Hadoop 설치는 완료입니다.
- ✓ 간단한 WordCount 예제를 Hadoop의 분산 처리, MapReduce를 이용하여 해결해보겠습니다.
- ✓ `B-9` 단계는 Namenode에서 진행됩니다. (Datanodes도 실행은 되어 있어야 합니다.)

```
# cd /usr/local/hadoop/bin  
# vi WordCount.java
```

다음 슬라이드의 내용을 `WordCount.java`에 입력하고 저장합니다.

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {
    public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable>{
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();
        public void map(Object key, Text value, Context context ) throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }
    public static class IntSumReducer extends Reducer<Text,IntWritable,Text,IntWritable> {
        private IntWritable result = new IntWritable();
        public void reduce(Text key, Iterable<IntWritable> values, Context context ) throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) { sum += val.get(); }
            result.set(sum);
            context.write(key, result);
        }
    }
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "word count");
        job.setJarByClass(WordCount.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

B-9. WordCount 예제 실행

```
# vi /usr/local/hadoop/etc/hadoop/hadoop-env.sh
```

```
HADOOP_CLASSPATH=/usr/local/jdk/lib/tools.jar
```

위 1행을 추가하고 저장합니다.

```
# cd /usr/local/hadoop/bin  
# hadoop com.sun.tools.javac.Main WordCount.java  
# jar cf wc.jar WordCount*.class
```

WordCount.java 파일을 컴파일해줍니다. wc.jar 파일도 생성합니다.

```
[root@namenode bin]# ls  
WordCount$IntSumReducer.class  hadoop      mapred.cmd  
WordCount$TokenizerMapper.class  hadoop.cmd  oom-listener  
WordCount.class                hdfs        test-container-executor  
WordCount.java                 hdfs.cmd    yarn  
container-executor              mapred       yarn.cmd  
[root@namenode bin]# jar cf wc.jar WordCount*.class  
[root@namenode bin]# ls  
WordCount$IntSumReducer.class  hadoop.cmd  test-container-executor  
WordCount$TokenizerMapper.class  hdfs        wc.jar  
WordCount.class                hdfs.cmd    yarn  
WordCount.java                 mapred       yarn.cmd  
container-executor              mapred.cmd  
hadoop                          oom-listener
```

B-9. WordCount 예제 실행

```
# hdfs dfs -mkdir user
# hdfs dfs -mkdir user/hadoop
# hdfs dfs -mkdir user/hadoop/wordcount
# hdfs dfs -mkdir user/hadoop/wordcount/input
# hdfs dfs -mkdir user/hadoop/wordcount/output
```

```
# vi file01
Hello World Bye World
# vi file02
Hello Hadoop Bye Hadoop
```

file01, file02를 생성하고 내용은 위와 같이 지정합니다.

```
# hadoop fs -copyFromLocal ./file0* /user/hadoop/wordcount/input
```

```
[root@namenode hadoop]# bin/hadoop fs -copyFromLocal ./file0* /user/hadoop/wordcount/input
```

```
2020-05-05 18:08:24,269 INFO sasl.SaslDataTransferClient: SASL encryption trust check:
localHostTrusted = false, remoteHostTrusted = false
2020-05-05 18:08:27,650 INFO sasl.SaslDataTransferClient: SASL encryption trust check:
localHostTrusted = false, remoteHostTrusted = false
```

위와 같이 출력되면 file01, file02가 HDFS 서버에 들어간 것입니다.

B-9. WordCount 예제 실행

```
# hdfs dfs -cat /user/hadoop/wordcount/input/file01  
# hdfs dfs -cat /user/hadoop/wordcount/input/file02
```

```
[root@namenode hadoop]# bin/hdfs dfs -cat /user/hadoop/wordcount/input/file01  
2020-05-05 18:09:04,502 INFO sasl.SaslDataTransferClient: SASL encryption trust check:  
localhostTrusted = false, remoteHostTrusted = false  
Hello World Bye World  
[root@namenode hadoop]# bin/hdfs dfs -cat /user/hadoop/wordcount/input/file02  
2020-05-05 18:09:13,596 INFO sasl.SaslDataTransferClient: SASL encryption trust check:  
localhostTrusted = false, remoteHostTrusted = false  
Hello Hadoop Goodbye Hadoop
```

HDFS 서버의 file01, file02를 cat으로 읽어보면 위와 같이 출력됩니다.

```
# hadoop jar wc.jar WordCount /user/hadoop/wordcount/input  
/user/hadoop/wordcount/output
```

```
[root@namenode hadoop]# bin/hadoop jar bin/wc.jar WordCount /user/hadoop/wordcount/inputs  
/user/hadoop/wordcount/output
```

HDFS 서버에 있는 file01, file02를 읽고 단어의 개수를 세어 /user/hadoop/wordcount/output으로 결과를 출력합니다.

B-9. WordCount 예제 실행

```
# hdfs dfs -cat /user/hadoop/wordcount/output/part*
```

```
[root@namenode hadoop]# bin/hdfs dfs -cat /user/hadoop/wordcount/output/part*
2020-05-05 18:10:49,096 INFO sasl.SaslDataTransferClient: SASL encryption trust check:
localHostTrusted = false, remoteHostTrusted = false
Bye 1
Goodbye 1
Hadoop 2
Hello 2
World 2
```

출력된 결과물(/user/hadoop/wordcount/output/에 있는)을 cat으로 읽습니다.

위와 같이 각 단어의 개수가 잘 세어졌다면 `B-9. WordCount 예제`까지 완료입니다.

부록 B. Install Hadoop 은 여기까지 입니다. 고생하셨습니다.



부록 C. Install R & R-studio in CentOS 7

- ✓ 부록 C. Install R & R-studio in CentOS 7은 부록 B에서 무사히 Hadoop 설치 및 설정을 마쳤다는 가정 하에 진행됩니다.
- ✓ 본 과정은 부록 B에서 **Namenode**로 동작하는 머신에서 진행됩니다.

```
# yum install epel-release
```

```
[root@namenode ~]# yum install epel-release
```

EPEL(Extra Packages for Enterprise Linux)은 Fedora Project에서 제공되는 저장소로 각종 패키지의 최신 버전을 제공하는 community 기반의 저장소입니다. R을 설치하기 위해 먼저 EPEL은 설치합니다.

```
# yum update
```

Epel을 적용시켜 주기 위해 yum을 업데이트 합니다.

```
# yum install R -y
```

R을 설치합니다.

C-1. Install R

R

```
[root@namenode ~]# R

R version 3.6.0 (2019-04-26) -- "Planting of a Tree"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-redhat-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

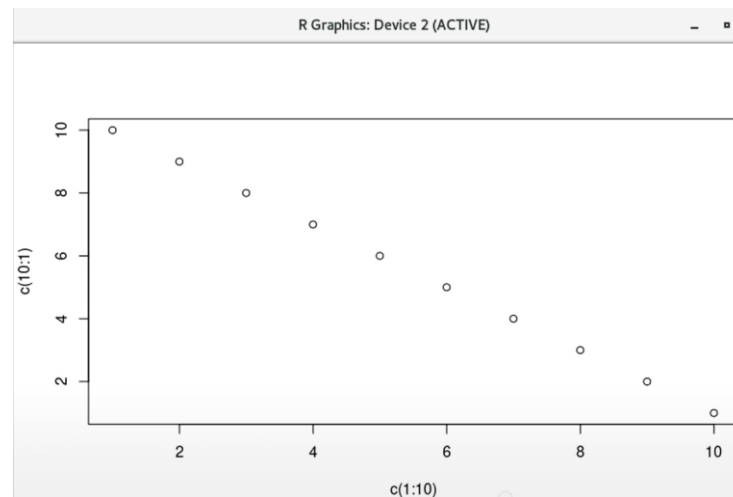
R을 실행시켰을 때 위와 같이 작동하면 R이 정상적으로 설치된 것입니다.

C-1. Install R

```
> plot(x=c(1:10), y=c(10:1));  
> c(1:100);
```

```
> plot(x=c(1:10),y=c(10:1));  
> c(1:100);  
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18  
[19] 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36  
[37] 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54  
[55] 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72  
[73] 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90  
[91] 91 92 93 94 95 96 97 98 99 100
```

시험삼아 R 코드를 작성해봅니다. (정상적으로 작동하는 것을 보실 수 있습니다.)



C-2. Install R-studio

- ✓ 이번에는 R을 편하게 이용할 수 있도록 만들어진 IDE인 R-studio를 설치해보겠습니다.
- ✓ 리눅스에서는 윈도우와 달리 R-studio를 하나의 소프트웨어처럼 사용할 수 없고 서버를 가동시켜 웹 브라우저를 이용해야 한다는 점에 유의해주시기 바랍니다.

```
# wget https://download2.rstudio.org/server/centos6/x86_64/rstudio-server-rhel-1.2.5042-x86_64.rpm sudo yum install rstudio-server-rhel-1.2.5042-x86_64.rpm
```

wget으로 R-studio package의 rpm파일을 다운로드 받습니다.

```
[root@namenode ~]# ls
서식  음악  바탕화면  hadoop-3.2.1.tar.gz
공개  사진  다운로드  initial-setup-ks.cfg
문서  비디오  anaconda-ks.cfg  rstudio-server-rhel-1.2.5042-x86_64.rpm
[root@namenode ~]# yum install rstudio-server-rhel-1.2.5042-x86_64.rpm
```

위처럼 다운받은 rpm 파일이 존재하는 위치에서 해당 rpm 파일을 설치합니다.

```
# yum install rstudio-server-rhel-1.2.5042-x86_64.rpm
```

C-2. Install R-studio

```
# systemctl status rstudio-server.service
```

```
[root@namenode ~]# systemctl status rstudio-server.service
● rstudio-server.service - RStudio Server
   Loaded: loaded (/etc/systemd/system/rstudio-server.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2020-05-05 20:34:00 KST; 18s ago
   Process: 115100 ExecStart=/usr/lib/rstudio-server/bin/rserver (code=exited, status=0/SUCCESS)
   Main PID: 115101 (rserver)
     Tasks: 3
    CGroup: /system.slice/rstudio-server.service
            └─115101 /usr/lib/rstudio-server/bin/rserver

May 05 20:34:00 namenode systemd[1]: Starting RStudio Server...
May 05 20:34:00 namenode systemd[1]: Started RStudio Server.
```

systemctl로 rstudio-server.service의 상태를 확인합니다.

현재는 active 상태로 표시되는데, 이것이 꺼져 있을 때를 대비하여 아래 2개 명령어를 입력합니다.

```
# systemctl restart rstudio-server.service
# systemctl enable rstudio-server.service
```

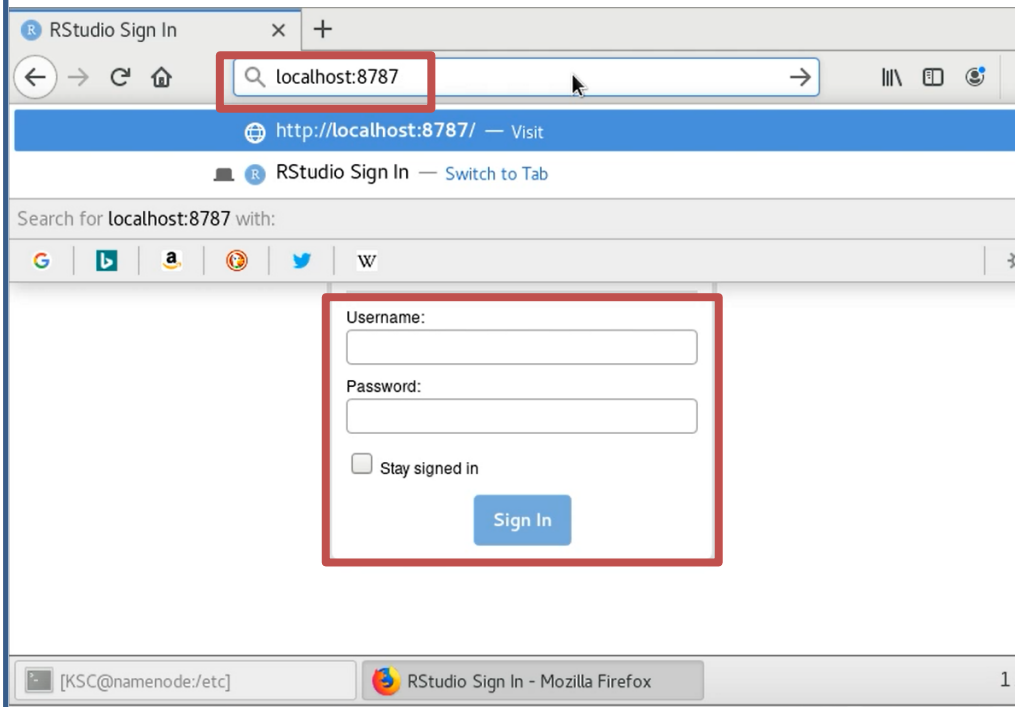
이제 R-studio를 사용하기 위한 작업은 모두 완료되었습니다.

C-3. ADD R-USER & USE R-studio

```
# useradd ruser  
# passwd ruser
```

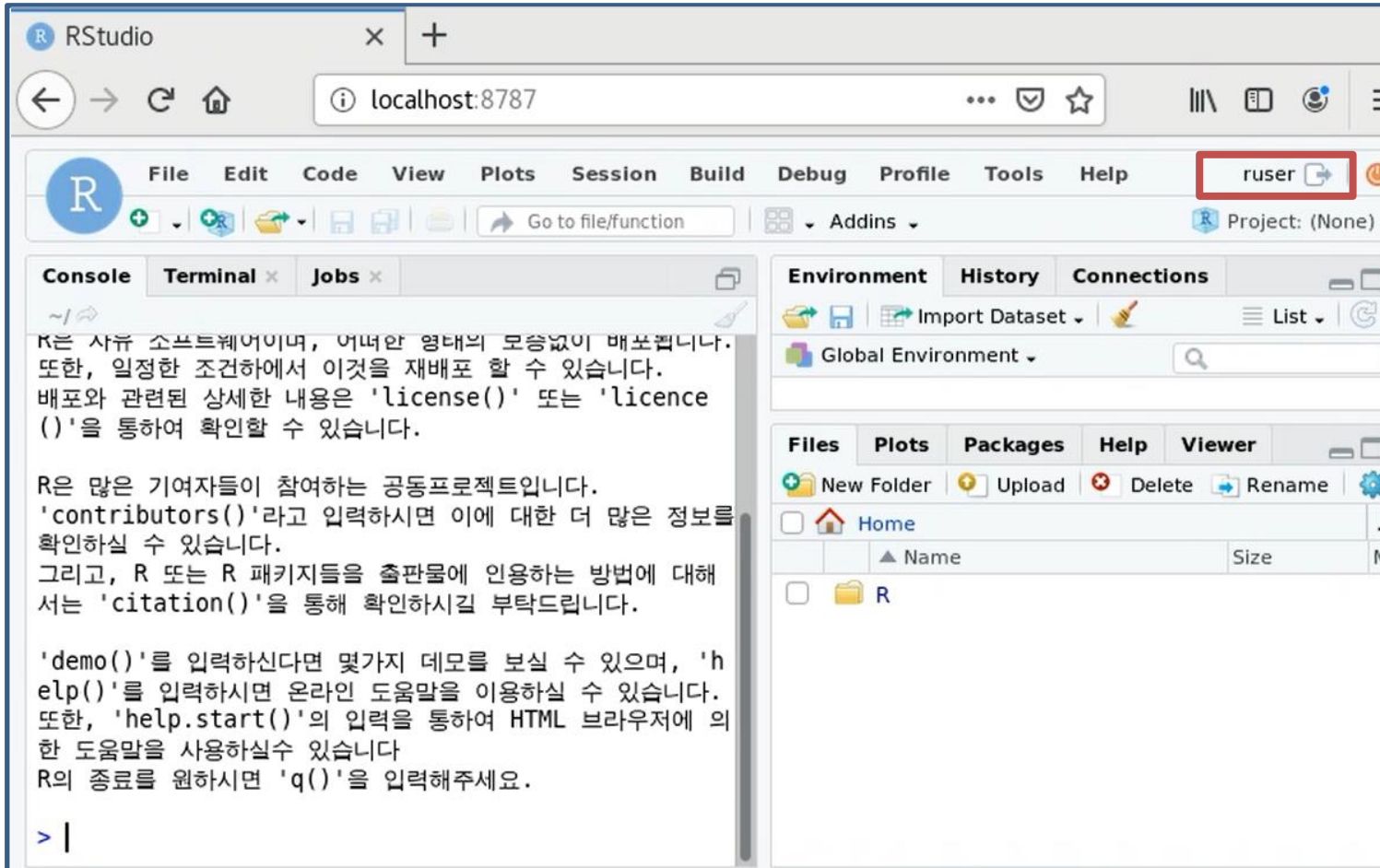
```
[root@namenode etc]# useradd ruser  
[root@namenode etc]# passwd ruser
```

R-studio를 사용하기 위한 사용자, ruser 계정을 생성하고, password를 설정합니다.



Firefox를 통해 localhost:8787/로 접속하면 sign in 페이지가 등장합니다.

C-3. ADD R-USER & USE R-studio



생성해 둔 ruser와 password로 접속하면 위와 같이 R-studio를 이용하실 수 있습니다.

C-3. ADD R-USER & USE R-studio

The screenshot shows the R Studio interface. The console pane on the left contains Korean text explaining the R environment and the 'c()' function. The environment pane on the right shows the 'Global Environment'. The 'Plots' pane on the right displays a scatter plot of the vector c(1:10).

Console Text:

```
( ) 을 통하여 확인할 수 있습니다.

R은 많은 기여자들이 참여하는 공동프로젝트입니다.
'contributors()'라고 입력하시면 이에 대한 더 많은 정보를
확인하실 수 있습니다.
그리고, R 또는 R 패키지들을 출판물에 인용하는 방법에 대해
서는 'citation()'을 통해 확인하시길 부탁드립니다.

'demo()'를 입력하신다면 몇가지 데모를 보실 수 있으며, 'h
elp()'를 입력하시면 온라인 도움말을 이용하실 수 있습니다.
또한, 'help.start()'의 입력을 통하여 HTML 브라우저에 의
한 도움말을 이용하실 수 있습니다
R의 종료를 원하시면 'q()'을 입력해주세요.

> c(1:10)
[1] 1 2 3 4 5 6 7 8 9 10
> plot(c(1:10));
```

Plots Pane:

The plot shows a scatter plot of the vector c(1:10). The y-axis is labeled 'c(1:10)' and has a tick mark at 2. The x-axis is labeled 'Index' and has tick marks at 2, 4, 6, 8, and 10. The plot displays 10 data points, each represented by a small circle, arranged in a slightly upward-sloping line.

간단한 예제 코드를 실행해 보며 R-studio 작동을 테스트합니다.

부록 C. Install R & R-studio 는 여기까지 입니다. 고생하셨습니다.