System Programming & OS 실습
3. Command Day 3

최민국

Dankook University

mgchoi@dankook.ac.kr

Index

1. 파일 및 디렉토리 관리

명령어	명령어 간단 설명	간단 예시
alias	명령어 별칭 설정	alias ll='ls -la'
cat	파일 내용 출력	cat file.txt
cd	디렉토리 이동	cd /home/user
cp	파일 또는 디렉토리 복사	cp file.txt backup.txt
echo	문자열 출력	echo 'Hello World'
export	환경 변수 설정	export PATH=$PATH:/new/path
ls	디렉토리 내 파일 목록을 출력	ls -l
mkdir	새로운 디렉토리 생성	mkdir new_folder
mv	파일 또는 디렉토리 이동/이름 변경	mv old.txt new.txt
pwd	현재 디렉토리 경로 출력	pwd
rm	파일 또는 디렉토리 삭제	rm file.txt
rmdir	빈 디렉토리 삭제	rmdir old_folder
touch	새로운 빈 파일 생성	touch newfile.txt

1.1. alias

- **alias란?**
  - 자주 사용하는 명령어를 단축하여 실행할 수 있도록 설정하는 기능
  - 긴 명령어나 반복 입력이 필요한 명령어를 간편하게 사용 가능
- **사용 목적**
  - 긴 명령어를 짧게 줄여 빠르게 실행
  - 특정 옵션을 항상 포함하여 실행
  - 실수 방지를 위해 삭제 명령어 등에 확인 옵션 추가
- **실습 예제**
  - $ alias ll='ls -la'
    - ll을 입력하면 ls -la 명령어가 실행되어 숨김 파일 포함 상세 목록 출력
  - $ alias rm='rm -i'
    - rm을 실행할 때 삭제 여부를 확인하도록 설정하여 실수 방지
- **주의사항**
  - 특정 alias 해제는 $ unalias 단축명 명령어로 가능

1.2. echo

- **echo란?**
  - 지정한 문자열 또는 변수 값을 출력하는 명령어
  - 터미널에서 메시지를 표시하거나 스크립트에서 값을 출력할 때 사용
- **사용 목적**
  - 간단한 텍스트 출력 및 메시지 표시
  - 환경 변수 값 확인 및 문자열 조작
  - 스크립트에서 실행 결과를 사용자에게 알릴 때 사용
- **실습 예제**
  - $ echo "Hello, Linux!"
    - "Hello, Linux!"라는 문자열을 출력
  - $ echo $HOME
    - 현재 사용자의 홈 디렉터리 경로를 출력
- **주의사항**
  - 작은따옴표(')와 큰따옴표(")에 따라 변수 해석 방식이 다를 수 있음
  - 환경 변수를 설정한 후 변경된 값을 즉시 반영하려면 export 명령어와 함께 사용 가능

1.3. export

- **export란?**
  - 현재 셸에서 환경 변수를 설정하여 하위 프로세스에서도 사용할 수 있도록 하는 명령어
  - 변수 값을 전역적으로 유지하여 프로그램이나 스크립트에서 활용 가능
- **사용 목적**
  - 특정 환경 변수를 설정하여 모든 하위 프로세스에서 접근 가능하도록 함
  - 실행 파일의 경로를 추가하여 명령어를 어디서든 실행할 수 있도록 설정
  - 스크립트 실행 시 필요한 변수를 미리 정의하여 환경 설정을 간편하게 관리
- **실습 예제**
  - $ export PATH=$PATH:/new/path
    - 기존 PATH 변수에 새로운 경로를 추가하여 해당 디렉터리의 실행 파일을 찾을 수 있도록 설정
  - $ export MY_VAR="Hello, Linux"
    - MY_VAR 변수에 값을 저장하여 현재 셸과 하위 프로세스에서 사용할 수 있도록 설정
- **주의사항**
  - export로 설정한 변수는 현재 셸과 하위 프로세스에서만 유효하며, 터미널을 종료하면 사라짐
  - 영구적으로 유지하려면 ~/.bashrc 또는 ~/.zshrc 파일에 추가 후 적용해야 함
    - echo 'export MY_VAR="Hello, World!"' >> ~/.bashrc
  - export된 변수 삭제는 $ unset 변수명 명령어를 사용

2. 파일 권한 및 소유권 관리

명령어	명령어 간단 설명	간단 예시
chmod	파일 권한 변경	chmod 755 script.sh
chown	파일 소유권 변경	chown user:user file.txt
chgrp	파일 그룹 변경	chgrp group file.txt

2.1. chmod

- **chmod란?**
  - 파일이나 디렉토리의 접근 권한을 변경하는 명령어
  - 소유자, 그룹, 기타 사용자에 대한 읽기, 쓰기, 실행 권한을 설정 가능
- **사용 목적**
  - 특정 사용자에게 파일 접근 권한을 부여하거나 제한할 때
  - 실행 파일의 실행 권한을 추가할 때
  - 보안 강화를 위해 중요한 파일의 접근을 제한할 때
- **실습 예제**
  - $ chmod 755 script.sh
    - 소유자는 읽기, 쓰기, 실행 가능하며 그룹과 다른 사용자는 읽기, 실행만 가능
  - $ chmod +x program.sh
    - program.sh 파일에 실행 권한을 추가하여 실행 가능하도록 설정
- **주의사항**
  - 권한을 잘못 변경하면 중요한 파일이 실행되지 않거나 보안 취약점이 발생할 수 있음
  - 루트 사용자로 chmod -R 777 / 같은 명령어 실행 금지 (시스템 보안 위험)

2.2. chown

- **chown란?**
  - 파일이나 디렉토리의 소유자 및 소유 그룹을 변경하는 명령어
  - 시스템 관리자가 파일 접근 권한을 조정할 때 사용
- **사용 목적**
  - 파일 소유권을 다른 사용자나 그룹으로 변경
  - 특정 사용자가 파일을 읽거나 수정할 수 있도록 권한 관리
- **실습 예제**
  - $ chown user1 file.txt
    - file.txt의 소유자를 user1으로 변경
  - $ chown user1:group1 file.txt
    - file.txt의 소유자와 그룹을 각각 user1과 group1으로 변경
- **주의사항**
  - 소유권 변경을 위해 관리자 권한이 필요하므로 sudo와 함께 사용해야 함
  - 잘못된 사용자 또는 그룹을 지정하면 접근 불가 문제가 발생할 수 있음

2.3. chgrp

- **chgrp란?**
  - 파일이나 디렉토리의 그룹 소유권을 변경하는 명령어
  - 파일을 특정 그룹이 관리할 수 있도록 설정 가능

- **사용 목적**
  - 다수의 사용자가 속한 그룹에서 특정 파일을 공동 관리할 때
  - 그룹 소유권을 변경하여 접근 권한을 조정할 때

- **실습 예제**
  - $ chgrp developers file.txt
    - file.txt의 그룹 소유권을 developers 그룹으로 변경
  - $ chgrp -R team project/
    - project 디렉토리 및 내부 파일들의 그룹을 team으로 변경

- **주의사항**
  - 그룹을 변경하려면 해당 파일에 대한 적절한 권한이 필요함
  - -R 옵션을 사용하면 하위 디렉토리와 파일까지 변경되므로 주의해야 함

3. 사용자 및 그룹 관리

명령어	명령어 간단 설명	간단 예시
useradd	새 사용자 추가	useradd newuser
userdel	사용자 삭제	userdel olduser
usermod	사용자 정보 수정	usermod -aG sudo user
groupadd	새 그룹 생성	groupadd newgroup
groupdel	그룹 삭제	groupdel oldgroup
whoami	현재 로그인한 사용자 출력	whoami

3.1. useradd

- **useradd란?**
  - 새 사용자를 시스템에 추가하는 명령어
  - 사용자 계정 생성 시 홈 디렉토리와 기본 설정을 함께 생성 가능
- **사용 목적**
  - 새로운 사용자 계정을 생성하여 시스템 접근 권한을 부여할 때 사용
  - 다중 사용자 환경에서 개별 계정을 관리하기 위해 필요
  - 보안 및 접근 제어를 위해 특정 사용자에게 개별 계정 제공
- **실습 예제**
  - $ sudo useradd newuser
    - newuser라는 새로운 사용자 계정을 생성
  - $ sudo useradd -m -s /bin/bash user1
    - user1 계정을 생성하고 홈 디렉토리와 기본 셸을 /bin/bash로 설정
- **주의사항**
  - useradd 명령어로 생성한 계정은 기본적으로 비밀번호가 설정되지 않음
  - 비밀번호를 설정하려면 $ sudo passwd newuser 명령어를 사용해야 함
  - 사용자 계정 삭제는 $ sudo userdel -r username 명령어를 사용하여 홈 디렉토리까지 삭제 가능

3.2. userdel

- **userdel이란?**
  - 시스템에서 기존 사용자 계정을 삭제하는 명령어
  - 사용자의 홈 디렉토리 및 관련 파일을 함께 삭제할 수 있음
- **사용 목적**
  - 더 이상 필요하지 않은 사용자 계정을 제거할 때 사용
  - 보안 관리 및 접근 제한을 위해 불필요한 계정을 삭제할 때 필요
  - 특정 사용자 제거 시 해당 계정의 파일과 설정을 함께 삭제 가능
- **실습 예제**
  - $ sudo userdel username
    - username 계정을 삭제하지만 홈 디렉토리는 유지
  - $ sudo userdel -r username
    - username 계정을 삭제하고 홈 디렉토리 및 관련 파일도 함께 삭제
- **주의사항**
  - userdel 실행 후에도 사용자의 프로세스가 실행 중이면 삭제되지 않을 수 있음
  - 홈 디렉토리까지 삭제하려면 -r 옵션을 함께 사용해야 함
  - 삭제된 계정은 복구할 수 없으므로 신중하게 실행해야 함

3.3. usermod

- **usermod란?**
  - 기존 사용자 계정의 정보를 변경하는 명령어
  - 사용자 그룹, 홈 디렉토리, 로그인 셸 등의 속성을 수정 가능

- **사용 목적**
  - 사용자 계정의 정보를 업데이트할 때 사용
  - 특정 사용자를 새로운 그룹에 추가하거나 기본 그룹을 변경할 때 필요
  - 사용자 계정의 홈 디렉토리나 로그인 셸을 변경할 때 사용

- **실습 예제**
  - $ sudo usermod -aG sudo username
    - username 사용자를 sudo 그룹에 추가
  - $ sudo usermod -d /home/newhome -m username
    - username의 홈 디렉토리를 /home/newhome으로 변경하고 기존 파일을 이동

- **주의사항**
  - 그룹을 변경할 때 -aG 옵션을 사용하지 않으면 기존 그룹이 제거될 수 있음
  - 로그인 셸을 변경할 경우, 올바른 경로인지 확인해야 시스템 로그인 오류를 방지 가능
  - 사용자 이름을 변경하려면 $ sudo usermod -l newname oldname 명령어를 사용해야 함

3.4. groupadd

- **groupadd란?**
  - 새 사용자 그룹을 생성하는 명령어
  - 시스템 내에서 사용자 권한과 접근 제어를 관리하는 데 사용됨
- **사용 목적**
  - 특정 사용자들을 같은 그룹으로 묶어 권한을 일괄적으로 관리
  - 파일 및 디렉토리의 그룹 소유권을 설정하여 접근 제어 강화
  - 다중 사용자 환경에서 관리 효율성을 높이기 위해 사용
- **실습 예제**
  - $ sudo groupadd developers
    - developers라는 새 그룹을 생성
  - $ sudo groupadd -g 1001 admins
    - 그룹 ID가 1001인 admins 그룹을 생성
- **주의사항**
  - 기존에 존재하는 그룹과 동일한 이름으로 생성할 수 없
  - 특정 그룹을 삭제하려면 $ sudo groupdel 그룹명 명령어 사용
  - 그룹을 수정하려면 $ sudo groupmod 명령어를 활용하여 변경 가능

3.5. groupdel

- **groupdel란?**
  - 기존 그룹을 시스템에서 삭제하는 명령어
  - 그룹 삭제 시 해당 그룹에 속한 사용자의 계정은 영향을 받지 않음
- **사용 목적**
  - 더 이상 사용되지 않는 그룹을 제거하여 사용자 및 권한 관리를 정리할 때 사용
  - 특정 그룹의 역할이 불필요해졌을 때 보안 및 시스템 정리를 위해 삭제
- **실습 예제**
  - $ sudo groupdel developers
    - developers 그룹을 시스템에서 삭제
  - $ sudo groupdel testgroup
    - testgroup을 삭제하여 더 이상 시스템에서 사용되지 않도록 함
- **주의사항**
  - 그룹을 삭제하려면 해당 그룹이 현재 사용 중이지 않아야 함
  - 그룹이 사용자 계정의 기본 그룹일 경우 삭제가 불가능하므로, 해당 사용자의 기본 그룹을 변경해야 함
  - 삭제 전에 $ cat /etc/group 명령어로 현재 존재하는 그룹을 확인하는 것이 좋음

3.6. whoami

- **whoami란?**
  - 현재 터미널을 사용 중인 사용자의 계정 이름을 출력하는 명령어
  - 사용자가 누구인지 확인할 때 유용
- **사용 목적**
  - 현재 로그인한 사용자 이름을 확인할 때 사용
  - 권한 변경(sudo 등) 후 사용자가 변경되었는지 확인할 때 필요
  - 시스템 내 여러 계정을 사용할 때 현재 계정을 쉽게 식별
- **실습 예제**
  - $ whoami
    - 현재 로그인한 사용자 계정 이름을 출력
  - $ sudo whoami
    - root 권한으로 실행 시 현재 사용자가 root인지 확인
- **주의사항**
  - 현재 로그인한 사용자 계정만 출력하며 다른 사용자의 정보는 확인할 수 없음
  - 여러 사용자가 동시에 로그인한 경우, 개별 터미널에서 실행해야 각자의 계정 정보 확인 가능

4. 프로세스 관리

명령어	명령어 간단 설명	간단 예시
ps	현재 실행 중인 프로세스 출력	ps aux
pstree	프로세스를 트리 형태로 출력	pstree
kill	프로세스 종료	kill 1234
top	실시간 프로세스 모니터링	top
systemctl	시스템 서비스 관리	systemctl restart apache2

4.1. ps

- **ps란?**
  - 현재 실행 중인 프로세스 목록과 상태를 확인하는 명령어
  - PID(Process ID), CPU 사용량, 실행 시간 등의 정보를 출력
- **사용 목적**
  - 시스템에서 실행 중인 프로세스를 확인할 때 사용
  - 특정 프로세스를 찾아 종료하거나 상태를 분석할 때 유용
  - 프로세스가 정상적으로 실행되고 있는지 점검할 때 활용
- **실습 예제**
  - $ ps aux
    - 모든 사용자와 시스템 프로세스를 상세 정보와 함께 출력
  - $ ps -ef | grep apache
    - 실행 중인 프로세스 중 "apache" 관련 프로세스를 검색
- **주의사항**
  - 기본적으로 현재 사용자의 프로세스만 출력하므로, 전체 프로세스를 보려면 옵션을 추가해야 함
  - 실시간으로 프로세스를 확인하려면 ps 대신 top 또는 htop 사용이 더 유용할 수 있음

4.2. pstree

- **pstree란?**
  - 현재 실행 중인 프로세스를 트리 형태로 출력하는 명령어
  - 부모-자식 관계를 구조적으로 표시하여 프로세스 흐름을 쉽게 확인 가능
- **사용 목적**
  - 시스템에서 실행 중인 프로세스를 한눈에 파악할 때 사용
  - 특정 프로세스의 부모-자식 관계를 분석하여 문제 해결
  - 백그라운드에서 실행 중인 서비스나 데몬을 확인하는 데 유용
- **실습 예제**
  - $ pstree
    - 현재 실행 중인 모든 프로세스를 트리 형태로 출력
  - $ pstree -p
    - 각 프로세스의 PID(프로세스 ID)를 함께 출력
- **주의사항**
  - 프로세스가 많을 경우 출력이 길어질 수 있어 less와 함께 사용 가능 ($ pstree | less)
  - 특정 사용자만의 프로세스를 확인하려면 $ pstree username 옵션을 사용할 수 있음

4.3. kill

- **kill이란?**
  - 실행 중인 프로세스를 종료하는 명령어
  - 특정 프로세스 ID(PID)를 지정하여 종료 가능
- **사용 목적**
  - 응답하지 않는 프로그램이나 백그라운드에서 실행 중인 프로세스를 강제 종료할 때 사용
  - 시스템 리소스를 절약하거나 문제 해결을 위해 불필요한 프로세스를 중지할 때 활용
- **실습 예제**
  - $ kill 1234
    - PID 1234번 프로세스를 종료
  - $ kill -9 5678
    - PID 5678번 프로세스를 강제 종료(SIGKILL 신호 사용)
- **주의사항**
  - 프로세스를 종료하기 전에 반드시 PID를 확인해야 함 ($ ps aux 또는 $ top 사용)
  - -9 옵션(SIGKILL)은 프로세스를 강제로 종료하며, 데이터 손실 가능성이 있음

4.4. top

- **top란?**
  - 시스템에서 실행 중인 프로세스와 자원 사용량을 실시간으로 표시하는 명령어
  - CPU, 메모리 사용률 등을 모니터링할 수 있음
- **사용 목적**
  - 현재 실행 중인 프로세스를 확인하고 시스템 성능을 모니터링할 때 사용
  - CPU 및 메모리 사용량이 높은 프로세스를 찾아 관리할 때 필요
  - 특정 프로세스를 종료하거나 우선순위를 조정할 때 활용 가능
- **실습 예제**
  - $ top
    - 현재 실행 중인 프로세스 목록과 시스템 리소스 사용량을 실시간으로 표시
  - $ top -u username
    - 특정 사용자가 실행한 프로세스만 필터링하여 출력
- **주의사항**
  - top 실행 후 q 키를 누르면 종료할 수 있음
  - 특정 프로세스를 종료하려면 k를 누르고 종료할 프로세스 ID(PID)를 입력해야 함
  - 보다 상세한 프로세스 관리가 필요하면 htop 명령어를 사용할 수도 있음

4.5. top

- **systemctl이란?**
  - 시스템 서비스를 관리하는 명령어
  - 서비스의 시작, 중지, 상태 확인 등을 수행할 수 있음
- **사용 목적**
  - 시스템에서 실행 중인 서비스의 상태를 확인하고 관리
  - 특정 서비스를 수동으로 시작하거나 정지할 때 사용
  - 시스템 부팅 시 특정 서비스가 자동으로 실행되도록 설정 가능
- **실습 예제**
  - $ sudo systemctl start apache2
    - apache2 서비스를 시작
  - $ sudo systemctl enable ssh
    - 시스템 부팅 시 ssh 서비스가 자동으로 실행되도록 설정
- **주의사항**
  - 서비스를 관리하려면 관리자 권한이 필요하므로 sudo를 사용해야 함
  - 중요한 시스템 서비스 종료 시 시스템이 정상적으로 동작하지 않을 수 있음
  - 서비스 상태 확인은 $ systemctl status 서비스명 명령어로 가능

5. 네트워크 관련 명령어

명령어	명령어 간단 설명	간단 예시
ping	네트워크 연결 상태 확인	ping google.com
ifconfig	네트워크 인터페이스 정보 출력	ifconfig eth0
hostname	호스트 이름 확인/변경	hostname
netstat	네트워크 연결 상태 확인	netstat -tulnp
ssh	원격 서버 접속	ssh user@host

5.1. ping

- **ping이란?**
  - 네트워크 상태를 확인하기 위해 특정 호스트에 패킷을 전송하고 응답을 받는 명령어
  - 목적지 서버가 정상적으로 응답하는지 테스트하는 데 사용
- **사용 목적**
  - 네트워크 연결이 정상적으로 작동하는지 확인
  - 특정 서버 또는 장치까지의 응답 속도(지연 시간, 패킷 손실)를 측정
  - 인터넷 또는 로컬 네트워크 문제를 진단
- **실습 예제**
  - $ ping google.com
    - google.com 서버와의 연결 상태 및 응답 시간을 확인
  - $ ping -c 5 8.8.8.8
    - 8.8.8.8(구글 DNS)로 5개의 패킷을 전송하여 네트워크 상태를 점검
- **주의사항**
  - 기본적으로 ping 명령어는 무한히 실행되므로 중단하려면 Ctrl + C를 사용해야 함
  - 방화벽 설정에 따라 ping 요청이 차단될 수 있음
  - 서버의 과부하를 방지하기 위해 과도한 ping 요청은 지양해야 함

5.2. ifconfig

- **ifconfig란?**
  - 네트워크 인터페이스의 설정 및 상태 확인을 위한 명령어
  - IP 주소, 서브넷 마스크, MAC 주소 등을 확인하고 변경 가능
- **사용 목적**
  - 현재 네트워크 인터페이스의 설정과 상태를 확인할 때 사용
  - 특정 네트워크 인터페이스를 활성화 또는 비활성화할 때 필요
  - 네트워크 트러블슈팅 및 IP 설정 변경 시 활용
- **실습 예제**
  - $ ifconfig
    - 현재 시스템의 네트워크 인터페이스 상태 및 IP 주소 확인
  - $ ifconfig eth0 192.168.1.100 netmask 255.255.255.0
    - eth0 인터페이스에 IP 주소와 서브넷 마스크를 설정
- **주의사항**
  - 최신 Linux 배포판에서는 ifconfig 대신 ip 명령어 사용이 권장됨
  - 네트워크 설정을 영구적으로 변경하려면 수동으로 설정 파일을 수정해야 함
  - 관리자 권한이 필요하므로 sudo를 사용하여 실행해야 할 수 있음

5.3. hostname

- **hostname란?**
  - 현재 시스템의 호스트 이름을 확인하거나 변경하는 명령어 네트워크에서 컴퓨터를 식별하는 데 사용됨
- **사용 목적**
  - 시스템의 호스트 이름을 확인하거나 변경할 때 사용
  - 네트워크 환경에서 특정 시스템을 식별할 때 유용
  - 서버 환경에서 고유한 호스트 이름을 설정하여 관리할 때 활용
- **실습 예제**
  - $ hostname
    - 현재 시스템의 호스트 이름을 출력
  - $ sudo hostname new-hostname
    - 시스템의 호스트 이름을 'new-hostname'으로 변경
- **주의사항**
  - 영구적으로 변경하려면 /etc/hostname 파일을 수정하고 시스템을 재부팅해야 함
  - 네트워크 환경에서 호스트 이름을 변경하면 일부 서비스가 영향을 받을 수 있음

5.4. netstat

- **netstat란?**
  - 네트워크 연결, 라우팅 테이블, 인터페이스 상태 등을 확인하는 명령어
  - 시스템에서 현재 활성화된 네트워크 상태를 모니터링하는 데 사용
- **사용 목적**
  - 현재 열려 있는 네트워크 연결 및 포트를 확인
  - 네트워크 트래픽 및 통신 상태를 분석
  - 서버의 네트워크 서비스 및 포트 상태를 점검
- **실습 예제**
  - $ netstat -tulnp
    - 현재 실행 중인 프로세스와 관련된 모든 네트워크 포트 목록을 출력
  - $ netstat -an | grep ESTABLISHED
    - 현재 활성화된(연결된) 네트워크 연결을 필터링하여 출력
- **주의사항**
  - 최신 시스템에서는 netstat보다 ss 명령어 사용을 권장
  - 특정 포트만 확인하려면 $ netstat -an | grep : 포트번호 형식으로 검색 가능
  - 관리자 권한이 필요한 옵션(-p 등)을 사용할 때는 sudo를 함께 사용해야 함

5.5. ssh

- **ssh란?**
  - 원격 시스템에 안전하게 접속할 수 있도록 하는 명령어
  - 암호화된 연결을 통해 서버와 클라이언트 간 통신을 보장

- **사용 목적**
  - 원격 서버에 접속하여 관리 및 운영 작업 수행
  - 네트워크를 통해 파일을 안전하게 전송하거나 원격 명령 실행
  - 터미널 기반으로 원격 시스템을 안전하게 제어

- **실습 예제**
  - $ ssh user@remote_server
    - user 계정으로 remote_server에 SSH 접속
  - $ ssh -p 2222 user@remote_server
    - 기본 포트(22) 대신 2222번 포트를 사용하여 접속

- **주의사항**
  - 원격 서버에서 SSH 서비스(ssh daemon)가 실행 중이어야 접속 가능
  - 보안을 위해 기본 포트(22) 대신 다른 포트를 사용하거나 공개 키 인증을 설정하는 것이 권장됨
  - 비정상적인 다중 로그인 시도는 차단될 수 있으므로 반복된 실패를 피해야 함

6. 입출력 및 데이터 조작

명령어	명령어 간단 설명	간단 예시
<, >, >>	입출력 리디렉션	echo 'Hello' > file.txt
|	파이프(출력을 다음 명령어의 입력으로 사용)	ls -l | grep txt
grep	텍스트 검색	grep 'error' log.txt
tail	파일의 마지막 부분 출력	tail -n 10 file.txt

6.1. <, >, >>

- **<, >, >>란?**
  - 표준 입력과 출력을 파일과 연결하는 리디렉션 명령어
  - <는 파일을 입력으로 사용하고, >는 출력을 파일로 저장하며, >>는 기존 파일에 내용을 추가

- **사용 목적**
  - 명령어의 입력을 파일에서 가져오거나 출력을 파일로 저장할 때 사용
  - 기존 데이터를 보존하면서 새로운 출력을 추가하거나 덮어쓸 때 활용

- **실습 예제**
  - $ cat < input.txt
    - input.txt의 내용을 읽어 표준 출력(터미널)로 표시
  - $ echo "Hello, Linux!" > output.txt
    - output.txt 파일에 "Hello, Linux!"를 저장 (기존 내용은 삭제됨)
  - $ echo "Appending text" >> output.txt
    - output.txt 파일의 기존 내용을 유지하면서 "Appending text"를 추가

- **주의사항**
  - >를 사용할 경우 기존 파일 내용이 삭제되므로 주의해야 함
  - >>는 기존 파일에 데이터를 추가하므로 덮어쓰기를 방지할 수 있음
  - <는 표준 입력으로 사용할 파일이 존재해야 정상적으로 동작함

6.2. |

- **|란?**
  - 한 명령어의 출력을 다른 명령어의 입력으로 전달하는 기능
  - 여러 명령어를 조합하여 데이터 처리를 효율적으로 수행 가능

- **사용 목적**
  - 여러 명령어를 연결하여 복잡한 작업을 간단하게 수행
  - 텍스트 필터링, 검색, 정렬, 변환 등의 작업을 자동화
  - 명령어 조합을 통해 대량의 데이터를 효과적으로 처리

- **실습 예제**
  - $ ls -l | grep ".txt"
    - 현재 디렉토리에서 .txt 파일만 출력
  - $ cat file.txt | wc -l
    - file.txt의 총 줄 수를 계산

- **주의사항**
  - |를 사용할 때 앞 명령어의 출력이 다음 명령어의 입력 형식과 맞아야 함
  - 일부 명령어는 파이프 입력을 예상하지 않으므로 조합 시 테스트 필요

6.3. grep

- **grep란?**
  - 파일이나 출력된 문자열에서 특정 패턴을 검색하는 명령어
  - 정규식을 활용하여 강력한 검색 기능 제공
- **사용 목적**
  - 파일에서 특정 단어나 문장을 빠르게 찾을 때 사용
  - 명령어 출력 결과에서 필요한 정보만 필터링할 때 유용
  - 로그 파일 분석 및 대량의 텍스트 데이터에서 특정 패턴 검색 가능
- **실습 예제**
  - $ grep "error" logfile.txt
    - Logfile.txt에서 "error"라는 단어가 포함된 모든 줄을 출력
  - $ ps aux | grep "bash"
    - 실행 중인 프로세스 목록에서 "bash"가 포함된 항목을 검색
- **주의사항**
  - 대소문자를 구분하여 검색하므로 대소문자를 무시하려면 -i 옵션 사용 ($ grep -i "error" logfile.txt)
  - 특정 단어가 포함되지 않은 줄을 검색하려면 -v 옵션 사용 ($ grep -v "warning" logfile.txt)

6.4. tail

- **tail이란?**
  - 파일의 마지막 몇 줄을 출력하는 명령어
  - 기본적으로 마지막 10줄을 표시하며, 실시간 로그 모니터링에도 사용 가능
- **사용 목적**
  - 긴 파일에서 최근 변경된 내용을 빠르게 확인할 때 사용
  - 로그 파일을 실시간으로 모니터링하여 시스템 상태를 점검할 때 유용
  - 특정 개수의 마지막 줄만 출력하여 효율적으로 데이터 분석 가능
- **실습 예제**
  - $ tail /var/log/syslog
    - syslog 파일의 마지막 10줄을 출력
  - $ tail -n 20 error.log
    - error.log 파일의 마지막 20줄을 출력
- **주의사항**
  - 파일이 너무 크면 출력 시간이 길어질 수 있음
  - 실시간 모니터링 시 $ tail -f filename 옵션을 사용하면 파일이 업데이트될 때마다 자동 출력됨

7. 압축 및 아카이브

명령어	명령어 간단 설명	간단 예시
tar	파일 압축 및 해제	tar -cvf archive.tar file.txt
gunzip	gzip 압축 해제	gunzip file.gz

7.1. tar

- **tar란?**
  - 여러 파일을 하나의 아카이브 파일로 묶거나 압축을 해제하는 명령어
  - 파일 백업 및 배포 시 주로 사용됨
- **사용 목적**
  - 여러 개의 파일을 하나의 압축된 아카이브 파일로 묶을 때 사용
  - 백업 및 파일 전송을 쉽게 하기 위해 활용
  - 특정 파일이나 디렉토리를 압축하거나 압축을 해제할 때 유용
- **실습 예제**
  - $ tar -cvf archive.tar file1.txt file2.txt
    - file1.txt와 file2.txt를 archive.tar 파일로 묶음
  - $ tar -xvf archive.tar
    - archive.tar의 내용을 현재 디렉토리에 압축 해제
- **주의사항**
  - tar 자체는 압축 기능이 없으며 gzip(-z) 또는 bzip2(-j) 옵션과 함께 사용해야 함
  - 기존 파일과 충돌을 방지하기 위해 압축 해제 전 파일 목록을 확인하는 것이 좋음
  - 백업할 때 원본 파일을 덮어쓰지 않도록 주의해야 함

7.2. gunzip

- **gunzip란?**
  - .gz 확장자로 압축된 파일을 해제하는 명령어
  - gzip으로 압축된 파일을 원래 상태로 복원할 때 사용
- **사용 목적**
  - gzip으로 압축된 파일을 해제하여 원본 파일을 복원할 때 사용
  - 대용량 파일을 압축 해제하여 분석하거나 실행할 때 필요
- **실습 예제**
  - $ gunzip file.gz
    - file.gz 파일을 압축 해제하여 file로 복원
  - $ gunzip -k file.gz
    - file.gz 파일을 압축 해제하되 원본 압축 파일을 유지
- **주의사항**
  - 기본적으로 gunzip은 원본 .gz 파일을 삭제하며 해제된 파일만 남김
  - 원본 파일을 유지하려면 -k 옵션을 사용해야 함
  - 여러 개의 .gz 파일을 한 번에 해제하려면 $ gunzip *.gz 명령어 사용 가능

8. 디스크 및 메모리 관리

명령어	명령어 간단 설명	간단 예시
df	디스크 사용량 확인	df -h
du	디렉토리 크기 확인	du -sh folder
ipcs	IPC 리소스 상태 확인	ipcs
ipcrm	IPC 리소스 삭제	ipcrm -m 1234

8.1. df

- **df란?**
  - 파일 시스템의 디스크 사용량을 확인하는 명령어
  - 각 파일 시스템의 전체, 사용 중, 남은 공간을 표시
- **사용 목적**
  - 시스템의 디스크 공간이 얼마나 남았는지 확인할 때 사용
  - 특정 파티션이나 디스크의 사용률을 모니터링하기 위해 필요
  - 서버 운영 및 스토리지 관리 시 필수적으로 활용
- **실습 예제**
  - $ df
    - 시스템의 모든 파일 시스템의 디스크 사용량을 기본 형식으로 출력
  - $ df -h
    - 사람이 읽기 쉬운 단위(GB, MB)로 디스크 사용량을 표시
- **주의사항**
  - 기본 출력값은 블록 단위로 표시되므로 -h 옵션을 사용하여 가독성을 높이는 것이 좋음
  - 특정 디렉토리의 사용량을 확인하려면 $ df -h /경로 형식으로 실행 가능

8.2. du

- **du란?**
  - 파일 및 디렉토리의 디스크 사용량을 확인하는 명령어
  - 특정 파일이나 폴더가 차지하는 공간을 측정할 수 있음

- **사용 목적**
  - 디스크 공간 관리 및 불필요한 파일 정리
  - 특정 디렉토리의 크기를 확인하여 저장 공간 최적화
  - 시스템 성능 유지 및 용량 부족 문제 해결

- **실습 예제**
  - $ du -h filename
    - filename의 크기를 사람이 읽기 쉬운 형식으로 출력
  - $ du -sh /home/user
    - /home/user 디렉토리의 총 사용량을 요약하여 출력

- **주의사항**
  - 기본적으로 하위 디렉토리별 사용량을 출력하므로 필요에 따라 -s 옵션 사용
  - 대량의 파일이 포함된 디렉토리는 분석에 시간이 걸릴 수 있음
  - 루트 권한이 필요한 디렉토리는 sudo를 사용하여 실행해야 함

8.3. ipcs

- **ipcs란?**
  - 현재 시스템에서 사용 중인 IPC(Inter-Process Communication) 리소스를 확인하는 명령어
  - 공유 메모리, 메시지 큐, 세마포어 등의 정보를 출력
- **사용 목적**
  - 시스템에서 사용 중인 공유 메모리, 메시지 큐, 세마포어 상태를 확인
  - 특정 프로세스가 IPC 리소스를 정상적으로 해제하지 않았는지 점검
  - 프로세스 간 통신을 관리하고 문제 해결 시 활용
- **실습 예제**
  - $ ipcs -m
    - 현재 시스템에서 사용 중인 공유 메모리 목록을 출력
  - $ ipcs -q
    - 메시지 큐의 상태와 정보를 확인
- **주의사항**
  - 일반 사용자는 모든 IPC 정보를 볼 수 없으며, 관리자 권한이 필요할 수 있음
  - 불필요한 IPC 리소스를 정리하려면 $ ipcrm 명령어를 함께 사용

8.4. ipcrm

- **ipcrm란?**
  - 프로세스 간 통신(IPC)에서 사용되는 공유 메모리, 세마포어, 메시지 큐를 제거하는 명령어
  - 시스템에서 불필요한 IPC 리소스를 해제할 때 사용
- **사용 목적**
  - 특정 프로세스가 종료된 후에도 남아 있는 공유 메모리, 세마포어, 메시지 큐를 삭제
  - 시스템 리소스를 정리하여 불필요한 메모리 점유 방지
  - 특정 IPC 리소스가 충돌할 경우 이를 수동으로 제거
- **실습 예제**
  - $ ipcrm -m 12345
    - 공유 메모리 세그먼트 ID 12345를 제거
  - $ ipcrm -s 67890
    - 세마포어 ID 67890을 제거
- **주의사항**
  - 제거할 대상 ID를 확인하려면 $ ipcs 명령어를 사용해야 함
  - 잘못된 ID를 제거하면 다른 프로세스의 동작에 영향을 줄 수 있으므로 주의 필요
  - 모든 IPC 리소스를 한 번에 삭제하려면 $ ipcrm --all 옵션을 사용하지만, 신중하게 실행해야 함

9. 쉘 관련 명령어

명령어	명령어 간단 설명	간단 예시
sh	쉘 스크립트 실행	sh script.sh
source	쉘 환경 설정 파일 실행	source ~/.bashrc
history	명령어 히스토리 확인	history
exit	쉘 종료	exit

9.1. sh

- **sh란?**
  - Bourne Shell(sh)을 실행하는 명령어
  - 스크립트를 실행하거나 기본 셸 환경을 변경하는 데 사용
- **사용 목적**
  - 셸 스크립트를 실행하여 자동화된 작업 수행
  - 다른 셸 환경에서 기본 Bourne Shell을 실행
- **실습 예제**
  - $ sh script.sh
    - script.sh 파일을 실행하여 스크립트 내용 수행
  - $ sh -c 'echo "Hello, Shell!"'
    - echo 명령어를 실행하여 "Hello, Shell!" 출력
- **주의사항**
  - 실행하려는 스크립트 파일에 실행 권한이 없으면 실행되지 않음
  - 실행 권한을 추가하려면 $ chmod +x script.sh 명령어를 사용해야 함
  - 특정 시스템에서는 sh가 기본적으로 다른 셸(bash, dash 등)로 연결될 수 있음

9.2. source

- **source란?**
  - 현재 쉘에서 스크립트나 설정 파일을 실행하고 적용하는 명령어
  - 새 프로세스를 생성하지 않고 환경 변수나 설정을 즉시 반영

- **사용 목적**
  - .bashrc 또는 .zshrc 같은 설정 파일을 적용할 때 사용
  - 스크립트 실행 후 환경 변수를 현재 쉘에 반영하기 위해 사용
  - 새로운 설정을 적용하기 위해 터미널을 다시 시작할 필요 없이 즉시 적용 가능

- **실습 예제**
  - $ source ~/.bashrc
    - 변경된 .bashrc 설정을 현재 터미널에서 즉시 적용
  - $ source script.sh
    - script.sh 파일을 실행하고 환경 변수를 현재 쉘에 반영

- **주의사항**
  - source를 사용하지 않고 스크립트를 실행하면 환경 변수 등이 현재 쉘에 반영되지 않음
  - 실행할 파일이 존재하지 않으면 "No such file or directory" 오류 발생

9.3. history

- **history란?**
  - 사용자가 실행한 명령어 기록을 확인하는 명령어
  - 이전에 실행한 명령어 목록을 저장하고 불러올 수 있음

- **사용 목적**
  - 최근 실행한 명령어를 확인하여 반복 작업을 쉽게 수행
  - 특정 명령어를 다시 실행하거나 수정하여 사용할 때 유용
  - 시스템 관리자가 사용자 활동을 추적할 때 활용

- **실습 예제**
  - $ history
    - 터미널에서 실행한 명령어 목록을 출력
  - $ history | grep "ls"
    - 실행한 명령어 중 "ls"가 포함된 기록만 출력

- **주의사항**
  - 기본적으로 사용자의 명령어 이력을 .bash_history 파일에 저장
  - history -c 명령어를 사용하면 전체 기록이 삭제됨
  - 중요하거나 민감한 명령어를 터미널에서 실행할 때 기록이 남는다는 점을 고려해야 함

9.4. exit

- **exit란?**
  - 현재 쉘(Session)이나 터미널을 종료하는 명령어
  - 종료 시 특정 종료 상태 코드를 반환할 수 있음
- **사용 목적**
  - 쉘 세션을 안전하게 종료할 때 사용
  - 스크립트 실행 후 특정 상태 코드를 반환하여 오류 처리에 활용 가능
- **실습 예제**
  - $ exit
    - 현재 터미널 또는 SSH 세션을 종료
  - $ exit 1
    - 종료 시 상태 코드 1을 반환하여 오류 발생을 나타냄
- **주의사항**
  - 실행 중인 프로세스가 있는 경우 종료 전에 저장 또는 백그라운드 실행 필요
  - SSH 원격 세션에서 실행하면 원격 접속이 종료됨

10. 관리자 권한 및 시스템 관리

명령어	명령어 간단 설명	간단 예시
su	다른 사용자 계정으로 전환	su - user
sudo	관리자 권한으로 명령 실행	sudo apt update
vi	텍스트 편집기 실행	vi file.txt

9.1. su

- **su란?**
  - 현재 사용자의 권한을 변경하거나 다른 사용자 계정으로 전환하는 명령어
  - 기본적으로 root 사용자로 전환하며, 특정 사용자로 변경도 가능

- **사용 목적**
  - 관리자(root) 계정으로 전환하여 시스템 설정을 변경할 때 사용
  - 특정 사용자 권한으로 작업을 수행해야 할 때 활용
  - 다중 사용자 환경에서 다른 계정으로 전환하여 보안 및 접근 제어 관리

- **실습 예제**
  - $ su
    - root 계정으로 전환 (root 비밀번호 필요)
  - $ su - username
    - username 계정으로 전환하여 해당 사용자의 환경 설정을 적용

- **주의사항**
  - root 계정으로 전환 시 실수로 시스템 설정을 변경할 위험이 있음
  - 권한이 필요한 작업에는 $ sudo를 사용하는 것이 보안적으로 더 안전함
  - su 실행 후 exit 명령어로 원래 계정으로 돌아올 수 있음

9.2. sudo

- **sudo란?**
  - 일반 사용자가 관리자(root) 권한을 임시로 얻어 명령을 실행할 수 있도록 하는 명령어
  - 보안성을 유지하면서 시스템 관리 작업을 수행할 수 있음
- **사용 목적**
  - 관리자(root) 권한이 필요한 명령어를 실행할 때 사용
  - 특정 사용자에게 일부 시스템 관리 권한을 부여할 때 필요
  - 중요 시스템 파일 수정이나 패키지 설치 작업을 수행할 때 사용
- **실습 예제**
  - $ sudo apt update
    - 시스템 패키지 목록을 최신 상태로 업데이트
  - $ sudo useradd newuser
    - 새로운 사용자 계정을 관리자 권한으로 생성
- **주의사항**
  - sudo 사용 시 주의하지 않으면 시스템 설정을 변경하거나 파일을 삭제할 위험이 있음
  - sudo 명령을 남용하지 않도록 최소한의 권한만 부여하는 것이 권장됨
  - sudo 권한이 없으면 관리자(root) 사용자가 /etc/sudoers 파일을 수정하여 권한을 부여해야 함