|  |
| --- |
| * Ch\_1 \_ 런레벨, 자동완성과 히스토리 * Ch\_2 \_ gedit, nano, vi 기본 사용법 * Ch\_3 \_ vi 비정상 종료, CD Rom 마운트 * Ch\_4 \_ 리눅스 기본 명령어 * Ch\_5 \_ 사용자와 그룹 및 관련 명령어 * Ch\_6 \_ 파일의 허가권과 소유권 * Ch\_7 \_ 명령어:링크 * Ch\_8 \_ 프로그램 설치를 위한 dpkg   서버 구축 시 필요한 개념과 명령어   * 종료하는 방법  1. 바탕화면의 [▼ 아이콘] > 컴퓨터끄기/로그아웃 > 컴퓨터 끄기 2. 터미널/콘솔에서 시스템 종료 명령 입력   Poweroff, shutdown -p now, halt-p, init 0   * 시스템 재부팅  1. 바탕화면의 [▼ 아이콘] > 컴퓨터끄기/로그아웃 > 다시 시작 2. 터미널/콘솔에서 시스템 재부팅 명령 입력   Shutdown -r now, reboot, init 6   * 로그아웃  1. 바탕화면의 [▼ 아이콘] > 컴퓨터끄기,로그아웃 > 로그아웃 2. 터미널/콘솔에서 시스템 종료 명령 입력   Logout 또는 exit   * 가상 콘솔 * ‘가상의 모니터’. 우분투는 총 6개의 가상 콘솔을 제공 * 각각의 가상 콘솔로 이동하는 단축키는 Ctrl + Alt + F2 ~ F7   (Ctrl + Alt + F2은 X윈도우 모드)     * 3번째 가상화면에서 shutdown -h실행 (+5 는 5분뒤에 진행함) * 3 번째 가상화면은 root로 접속한 상태      * 4번째 가상화면에서 종료 메시지 확인 * 4번째 가상화면은 일반사용자(ubuntu)로 접속한 상태. 3번째 가상화면에서 진행된 shutdown -h 메시지 확인 가능 * 런 레벨 (Runlever)   Init 명령어 뒤에 붙은 숫자를 런 레벨(Runlever) 이라고 부른다.   * 런 레벨 0: Power Off (종료 모드) * 런 레벨 1: Rescue (시스템 복구 모드) // 단일 사용자 모드 * 런 레벨 2: Multi-User // 사용하지 않음 * 런 레벨 3: Multi-User (텍스트 모드의 다중 사용자 모드) [현 server (b)] * 런 레벨 4: Multi-User // 사용하지 않음 * 런 레벨 5: Graphical (그래픽 모드의 다중 사용자 모드) // 사용하지 않음 [현 server] * 런 레벨 6: Reboot   런 레벨 모드를 확인하려면 /lib/systemd/system 디렉터리의 runlevel?.target 파일을 확인한다.  런레벨 3로 변경     * 런레벨 관련 파일 (바로가기)      * 현재 server는 graphical 그래픽 모드로 부팅되어있는 것을 확인      * Server를 그래픽 모드에서 텍스트 모드로 변환 * Default.targer 이 graphical.target에서 multi-user.target으로 변경 * Ls -l default.target 으로 확인가능 * Ln -sf /lib/systemd/system/multi-user.target /lib/systemd.system/default.target 으로 변경      * 재부팅 시 텍스트 모드로 (터미널모드) 로 부팅 * graphical 그래픽모드로 부팅 원할 시 Startx 입력      * server를 텍스트모드에서 그래픽모드로 다시 변환 후 리부팅   자동 완성과 히스토리   * 자동 완성이란 파일명의 일부만 입력한 후 tab키를 눌러 나머지 파일명을 자동으로 완성하는 기능   Ex. Cd /etc/NetworkManager/dispatcher.d/ 를 입력하려면  Cd /et[tab]ne[tab]di[tab]   * 자동 완성기능은 빠른 입력효과도 있지만 파일명이나 디렉터리가 틀리지 않고 정확하게 입력되는 효과도 있으므로 자주 사용됨 * 도스 키란 이전에 입력한 명령어를 상/하 화살표 키를 이용하여 다시 나타내는 기능      * History 로 지금까지 사용한 명령어들 확인가능 * 보안상 등 이유로 불필요할시 history -c 로 클리어 가능      * 자동완성 기능으로만 완성된 명령어 * Cd /e[tab]/sysy[tab]/ne[tab]   에디터 사용 (gedit, nano, vi)     * Vi 에디터 사용법 개요도      * Gedit 에디터 화면 * Terminal에서 gedit 명령어 입력으로 사용 가능 * 메모장과 같은 기능을 가지고 있음      * Gedit 에디터로 만들어진 txt파일      * Nano 나노 에디터 화면 * Gedit 에디터와는 다르게 xwindow가 없어도 사용 가능 (gedit에디터는 xwindow에서만 사용가능) * Nano 에디터에서 Ctrl + c 로 행 번호 파악 가능 (nano -c 로 에디터를 오픈하면 계속 행번호가 나타남)      * Nano 에디터로 만들어진 txt파일 (myfile2.txt)      * Vi 에디터 사용 화면      * Vi 에디터도 나노 에디터와 동일하게 xwindow가 없어도 사용 가능 * vi에디터 실행 뒤 명령모드에서 I 또는 a명령어로 입력모드로 전환 후 사용가능 * 문서 작성 후 esc 로 명령모드 이후 shift + ; ex모드 (라인 명령 모드) ex모드에서 저장(w) 종료 (q) 취소(c) 가능 * 작성 중 저장하지 않고 종료 (q!)   Vi 비정상 종료       * Vi 에디터를 사용 중 터미널을 닫았을 때 나오는 오류      * Vi 에디터로 생성한 new.txt 파일 이외에 .new.txt.swp라는 임시파일 발견 * 정상 종료 시 임시파일은 삭제되는데 비 정상 종료로 인해 임시파일이 삭제x        * 임시파일 삭제 후 vi에디터 정상 작동 확인   Vi 기능 요약   * 명령 모드 > 입력 모드   i | 현재 커서의 위치부터 입력 (i)  I | 현재 커서 줄의 맨 앞에서부터 입력 (Shift + I)  a | 현재 커서의 위치 다음 칸부터 입력 (A)  A | 현재 커서 줄의 맨 마지막부터 입력 (Shitf + A)   * 명령 모드에서 커서를 이동   H | 커서를 왼쪽으로 한 칸 이동 (←와 같은 의미, H)  J | 커서를 아래로 한 칸 이동 (↓와 같은 의미, J)  K | 커서를 위로 한 칸 이동 (↑와 같은 의미, K)  I | 커서를 오른쪽으로 한 칸 이동 (→와 같은 의미, I)   * 명령 모드에서 삭제, 복사, 붙여넣기   x | 현재 커서가 위치한 글자 삭제 (del과 같은 의미, X)  X | 현재 커서가 위치한 앞 글자 삭제 (backspace와 같은 의미, shift+X)  dd | 현재 커서의 행 삭제 (D 연속 두 번 입력)  숫자 DD | 현재 커서부터 숫자만큼의 행 삭제 (숫자 다음 D 연속 두 번 입력)  yy | 현재 커서가 있는 행을 복사 (Y 연속 두 번 입력)  숫자 yy | 현재 커서부터 숫자만큼의 행을 복사 (숫자 다음 yy연속 두 번 입력)  p | 복사한 내용을 현재 행 이후에 붙여 넣기 (p)  P | 복사한 내용을 현재 행 이전에 붙여 넣기 (shift + p)     * 나노 에디터 ‘set number’ 명령어로 행 나타냄      * Man 명령어를 사용한 도움말 출력 * Man ls 명령어로 ls에 대한 도움말을 출력한 상태 * 넘기기는 space 뒤로 이동은 b 나가기는 q   마운트와 CD/DVD, USB의 활용   * 물리적은 장치를 특정한 위치 (대개는 디렉터리) 에 연결시켜 주는 과정      * mount명령어로 장치에 연결되어 있는 하드웨어장치들 위치 확인 * 현 장치에 연결되어 있는 하드웨어장치들이 어느 장소에 연결이 되어있는지 나타내는 명령어   CD 마운트     * 우분투 설치 dvd삽입 * Player > Removable Devices > CD/DVD > Connect      * 자동으로 연결된 상태 * 그래픽 모드에서만 자동으로 연결되고, 오류도 많을 수 있어서 수동 연결 권장      * 수동 연결을 위해 기존 연결 끊기 * Umount /dev/cdrom 명령어      * 수동으로 dvd 연결 (읽기 전용으로 마운트됨) * ~~mkdir cdrom 명령어를 사용하여 ‘cdrom’ 디렉토리 생성~~ * media 폴더에서 cdrom 파일 생성 (mkdir /media/cdrom 명령어 사용) * mount /dev/cdrom /media/cdrom 명령어를 사용하여 마운트   dev는 device의 약자로 설정해 놓은 iso파일을 가리킴     * dvd에 생성된 파일들 확인 가능      * 기존 연결 Umount시 오류 발생 * Media/cdrom에 연결된 상태에서 umonunt를 시도하여 오류가 발생 * 다른 디렉토리로 이동하여 오류 해결 가능      * Cd 명령어로 디렉토리 변경 후 umount 명령어 실행 * Cd(change directory) 사용하여 오류 해결 * 이후 player > Removable devices > …에서 iso파일 제거하면 물리적으로도 제거완료 * 리눅스 기본 명령어 (1) * ls   Windows의 ‘dir’ 와 같은 역할로, 해당 디렉토리 안에 있는 파일의 목록을 나열  ex # ls /etc/systemd | [자세히 보기 ls -l] [숨김 파일도 보기 ls -a]  합쳐서 사용 가능 <ls - al> 확장명으로 찾기 <ls \*.txt> < txt파일만   * cd   디렉토리를 이동   * pwd   현재 디렉토리의 전체 경로를 출력   * rm   파일이나 디렉토리를 삭제     * Root 폴더 new.txt 파일을 삭제 * mount /dev/cdrom /media/cdrom 명령어를 사용하여 마운트   dev는 device의 약자로 설정해 놓은 iso파일을 가리킴     * cdrom 폴더를 rm명령어를 사용하여 삭제 진행 시 오류 발생 * 폴더는 rm명령어로 삭제 불가능 * Rm -r 명령어로 삭제 가능 * 리눅스 기본 명령어 (2) * cp   파일이나 디렉토리를 복사  예 ) #cp abc.txt abb.txt     * myfile1 파일을 file1.txt 이름으로 복사 * touch   크기가 0인 새 파일을 생성, 이미 존재하는 파일의 생성 날짜를 최신화   * Mv   파일과 디렉토리의 이름을 변경하거나 이동  예 ) mv abc.txt www.txt     * Myfile2.txt 파일을 file2.txt 파일로 이름 변경 * mkdir   새로운 디렉토리를 생성  예 ) #mkdiv abc   * 리눅스 기본 명령어 (3) * rmdir   디렉토리를 삭제. (단, 비어 있어야 함)  예) # rmdir abc   * Cat   텍스트로 작성된 파일을 화면에 출력  예) cat a.txt b.txt   * Head, tail   텍스트로 작성된 파일의 앞 10행 또는 마지막 10행 출력  예) #head /etc/systemd/bootchart.conf   * More   텍스트로 작성된 파일을 화면에 페이지 단위로 출력  예) # more /etc/systemd/system.conf   * 리눅스 기본 명령어 (4) * Less   More와 용도가 비슷하지만 더 확장됨  예) # less /etc/systemd/system.conf   * File   File이 어떤 종류의 파일인지를 표시  예) # file /etc/systemd/system.conf   * clear   명령창을 깨끗하게 지움  예) clear   * 사용자와 그룹 (1) * 리눅스는 다중 사용자 시스템(multi-user System) 임 * 기본적으로 root라는 이름을 가진 슈퍼유저(superuser)가 있으며, 모든 작업을 할 수 있는 권한이 있음 * 모든 사용자는 하나 이상의 그룹에 소속되어 있음 * 사용자는 /etc/passwd 파일에 정의되어 있음      * 각 행의 의미는 다음과 같음   사용자 이름:암호:사용자ID:사용자가 소속된 그룹ID:추가정보:홈 디렉터리:기본 셀   * 암호 x는 따로 관리하고 있음 (암호가 없다는 것이 아님) * Root 사용자의 홈 디렉터리는 /root   일반 사용자의 홈 디렉터리는 /home/사용자이름   * 사용자와 그룹 (2) * 사용자의 비밀번호는 /etc/shadow 파일에 정의되어 있음 * 그룹은 /etc/group 파일에 정의되어 있음      * 각 행의 의미는 다음과 같음   그룹명:비밀번호:그룹 id:보조 그룹 사용자 (보조 그룹 사용자는 참고사항)   * 보조 그룹 사용자 (root 기준 0) 은 passwd파일 사용자가 소속된 그룹ID(root 기준0) 과 같은 의미 * 사용자와 그룹 관련 명령어 (1)      * adduser   새로운 사용자를 추가  예) # adduser newuser1   * passwd   사용자의 비밀번호를 지정하거나 변경  예) # passwd newuser1   * usermod   사용자의 속성을 변경  예) # usermod –groups ubuntu newuser1   * userdel   사용자를 삭제  예) #user newuser2   * chage   사용자의 암호를 주기적으로 변경하도록 설정  예) # chage -m 2 newuser1   * groups   현재 사용자가 속한 그룹을 보여줌  예) # groups   * groupadd   새로운 그룹을 생성  예) # groupadd newgroup1   * groupmod   그룹의 속성을 변경  예) #grouomod – new-name mygroup1 newgroup2   * 사용자와 그룹 관련 명령어 (3) * groupdel   그룹을 삭제  예) # groupdel newgroup2   * gpasswd   그룹의 암호를 설정하거나, 그룹의 관리를 수행  예) # gpasswd mygoup1   * 실습      * user1 사용자를 추가 * adduser 명령어 입력 (adduser user1) * tail 명령어 사용하여 passwd파일 맨 마지막 생성된 user1 사용자 정보 확인      * 그룹을 따로 설정하지 않은 user1의 그룹 확인 * 그룹을 지정하지 않고 생성하게 되면 자동으로 그룹도 생성 (사용자 이름과 같음) 일반적인 상황에서는 그룹부터 생성하고 유저를 생성      * 사용자를 삭제. * -r 명령어를 추가하면 사용자가 사용하던 디렉터리까지 삭제      * 그룹 생성 뒤 해당 그룹 지정하여 사용자 생성 * groupadd 명령어를 사용하여 ubuntuGroup 그룹 생성 후 해당 그룹 번호 확인 * adduser 명령어로 사용자 생성. –-gid 명령어 사용하여 그룹 지정 (adduser -–gid 1001 user1) adduser –gid (그룹 번호) (사용자 name)      * 같은 방법으로 사용자2(user2) 생성 뒤 두 사용자의 정보 확인 * user1, user2모두 같은 그룹 (1001) 인 모습을 확인 가능      * shadow 파일을 통해 user의 비밀번호 정보 확인 * 비교를 위해 user1 과 user2의 비밀번호를 동일하게 설정(‘1234’) 하였으나 암호화된 비밀번호는 전혀 다른 패턴 (비밀번호 파악 유추 불가)      * 생성된 사용자들의 디렉터리 확인 * /home/user1 디렉터리의 파일들은 /etc/skel/ 디렉터리의 파일들을 복사해서 넣는 개념 (사용자를 생성할 때 추가할 파일들은 skel 디렉터리에 추가하면 됨)        * xwindow 모드로도 관리 가능 * 암호 설정 시 8글자 이상 , 문자+숫자 필수 * 파일과 디렉터리의 소유와 허가권 (1) * 파일의 리스트와 파일 속성      * 파일과 디렉터리의 소유와 허가권 (2) * 파일 유형   디렉터리일 경우에는 d, 일반적인 파일일 경우에는 -가 표시   * 파일 허가권 (Permission)   “rw-“, “r--“, “r--” 3개씩 끊어서 읽음 ( r은 read, w는 write, x는 execute의 약자)  첫 번째 “rw-“는 소유자(user)의 파일접근 권한  두 번째의 “r--“는 그룹(Group)의 파일접근 권한  세 번째의 “r--”는 그 외의 사용자(other)의 파일접근 권한  숫자로도 표시 가능 (8진수)     * 파일과 디렉터리의 소유와 허가권 (3) * chmod 명령   파일 허가권 변경 명령어  예) # chmod 777 sample.txt   * 파일 소유권(Ownership)   파일을 소유한 사용자와 그룹을 의미   * chown/chgrp 명령   파일의 소유권을 바꾸는 명령어 (root사용자만 사용가능)  예) # chown ubuntu.ubuntu sample.txt 또는  소유자 소유그룹  # chown ubuntu sample.txt 및 # chgrp ubuntu sample.txt     * 관리자 권한인 root 사용자로 test 파일 실행 시 허가 거부 발생 * nano 에디터로 test 파일 생성 * root 사용자로 nano 에디터로 생성된 test파일 실행 > 허가 거부 * root 사용자 권한이 rw(읽기 쓰기) 만 허용, 실행 권한 없는 상태      * 실행 권한 추가 후 실행 완료 * test파일의 파일 허가권을 755로 변경 (‘chmod 755 test’)   -rw-r--r-- > -rwx-r-x-r-x 로 변경됨   * 변경 후 root 사용자로 실행 완료      * ubuntu 사용자로 소유권 변경 * chown 명령어를 사용하여 test 파일의 소유권을 ubuntu 사용자로 변경 * chgrp 명령어를 사용하여 test 파일의 소유권을 ubuntu 그룹으로 변경 * 링크 * 파일의 링크(Link)에는 하드 링크(Hard Link)와 심볼릭 링크(Symbolic Link 또는 Soft Link) 두 가지가 있음      * 하드 링크를 생성하면 “하드 링크파일”만 하나 생성되며 같은 inode1을 사용 (명령: # In 링크대상파일이름 링크파일이름) * 심볼릭 링크를 생성하면 새로운 inode2를 만들고, 데이터는 원본 파일을 연결하는 효과 (명령 : # In -s 링크대상파일이름 링크파일이름) * linktest 디렉터리에 basefile과 basefile에 연결되는 hardlink, softlink 생성 * 연결 확인 후 basefile을 다른 디렉터리로 이동 (mv명령어)      * hardlink는 작동되나 softlink는 파일을 찾지 못함 * hardlink는 원본 파일과 같은 inode1에 연결하여 원본 파일 데이터와 연결된 반면 softlink는 inode2에 연결되어 원본 파일 포인터로 원본 파일을 찾아야 하기 때문에 다른 디렉터리로 이동한 원본 파일을 찾지 못함      * 원본 파일이 원래 디렉터리(linktest) 로 돌아오니 softlink도 정상 작동 * 프로그램 설치를 위한 dpkg (1) * dpkg (Debian Package)   windows의 “setup.exe”와 비슷한 설치 파일  확장명은 \*.deb이며, 이를 패키지(package)라고 부름.   * 파일의 의미     패키지이름\_버전-개정번호\_아키텍쳐.deb  패키지이름 : galculator > 패키지(프로그램)의 이름  버전 : 2.1.4 > 대개 3자리수로 구성. 주 버전,부 버전,패치버전  개정번호 : 1 > 문제점을 개선할 때마다 붙여지는 번호  아키텍처 : amd64 > 64비트 CPU를 의미     * 프로그램 설치를 위한 dpkg (2) * 자주 사용하는 dpkg 명령어 옵션 * 설치   dpkg -I 패키지파일이름.deb (소문자 아이)   * 삭제   dpkg -r 패키지이름  dpkg -p 패키지이름 > 설정파일까지 삭제   * 패키지 조회   dpkg -l 패키지이름 > 설치된 패키지에 대한 정보를 보여줌 (소문자 엘)  dpkg -L 패키지이름 > 패키지가 설치한 파일 목록을 보여줌   * 아직 설치되지 않은 deb 파일 조회   dpkg --info 패키지파일이름.deb > 패키지 파일에 대한 정보를 보여줌   * 프로그램 설치를 위한 dpkg (3) * dpkg 명령의 단점 * ‘의존성’ 문제   A패키지가 설치되기 위해서 B패키지가 필요할 경우, dpkg 명령으로는 해결이 까다로움   * 이름 해결하기 위해 apt 가 등장      * deb 파일 2개 저장 상태 확인      * “dpkg --info 패키지이름” 명령어로 해당 패키지 내용 확인      * “dpkg -I 패키지이름” 명령어로 해당 패키지 다운로드      * “dpkg -r 패키지이름” 명령어로 해당 패키지 삭제      * 의존성 패키지를 dpkg -I 명령어로 설치 시 오류 * 해당 패키지가 의존하는 패키지부터 설치해야 오류 해결 |