제1장 운영체제의 이해

1. SW의 종류 및 개념

1) System SW: 컴퓨터의 작동에 필수적인 SW(OS등)

2) Application SW: 특정 작업을 수행하기 위한 SW(주문형 SW, 상용 SW)

3) Open Source: 소스 코드를 무상으로 공개하여 누구나 재배포 가능한 SW

4) Freeware: 저작권에 관계없이 무료로 배포하는 SW

5) Shareware: 일정기간 사용한 후 돈을 지불해야하는 SW

6) Beta Version: 많은 사람들에 의해 공개적인 테스트를 진행하는 버전

7) Alpha Version: 회사 내부에서 테스트를 진행하는 버전

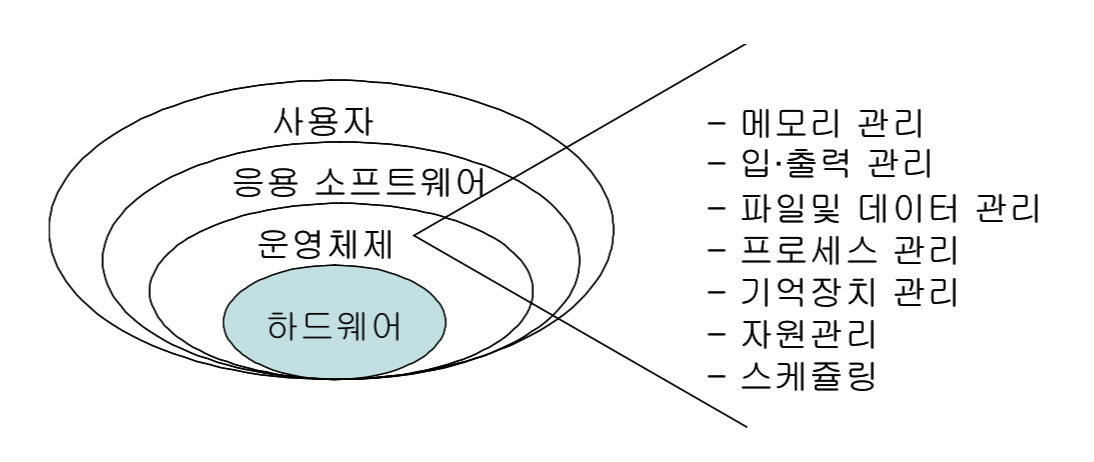
8) Trial Version: 일정기간 사용 후 구매

9) Bundled SW: 패키지의 일부로 함께 판매되는 SW

2. 운영체제란?

1) 컴퓨터를 작동 및 관리하고

2) 응용 프로그램이 효율적으로 실행될 수 있는 환경을 제공하는 SW



3. 운영체제의 기능

1) 메모리 관리: RAM 제어

2) 입,출력 관리: 키보드, 모니터 등 주변기기 제어

3) 파일 및 데이터 관리: 파일, 데이터를 유지하고 관리

4) 프로세스 관리

5) 기억장치 관리

6) 자원 관리

7) 스케줄링: 작업 순서를 효율적으로 관리

4. 운영체제가 가져야 할 조건

1) 처리능력 향상

2) 응답시간 단축

3) 사용가능성 향상

4) 신뢰도 향상

5. 운영체제의 유형

1) 다중 교환: 다수 작업이 동시에 수행되나, 포그라운드 프로그램만 동작하는 형태

2) 단일 작업: 한 번에 하나의 작업만 수행

3) 다중 작업: 여러 작업을 동시에 수행

4) 다중 사용자: 단일 프로세스에서 여러 사용자의 프로그램이 실행

5) 대화형 처리

6) 일괄 처리: 여러 개의 작업을 묶어 한꺼번에 처리

7) 실시간 처리

8) 가상 머신: 하나의 시스템에 여러 개의 작은 시스템이 있는 것처럼 분할하여 동작하게 해주는 SW

9) 분산 처리: 여러 시스템이 작업을 나누어 처리

10) 병렬 처리: 여러 개의 프로세서로 입출력, 연산 등을 동시에 처리

6. 운영체제의 종류

1) 개인용 OS(Windows, Mac OS 등)

2) 워크스테이션용 OS(Linux, Unix, Windows Server 등)

- 최소한 프로세스만 구동

- 데이터 유실, 장애 대책

- 오라클 등 SW 라이센스

7. Unix의 특징

1) 뛰어난 통신 기능

2) 복수 사용자와 다중 처리 지원

3) 뛰어난 호환성, 유연성

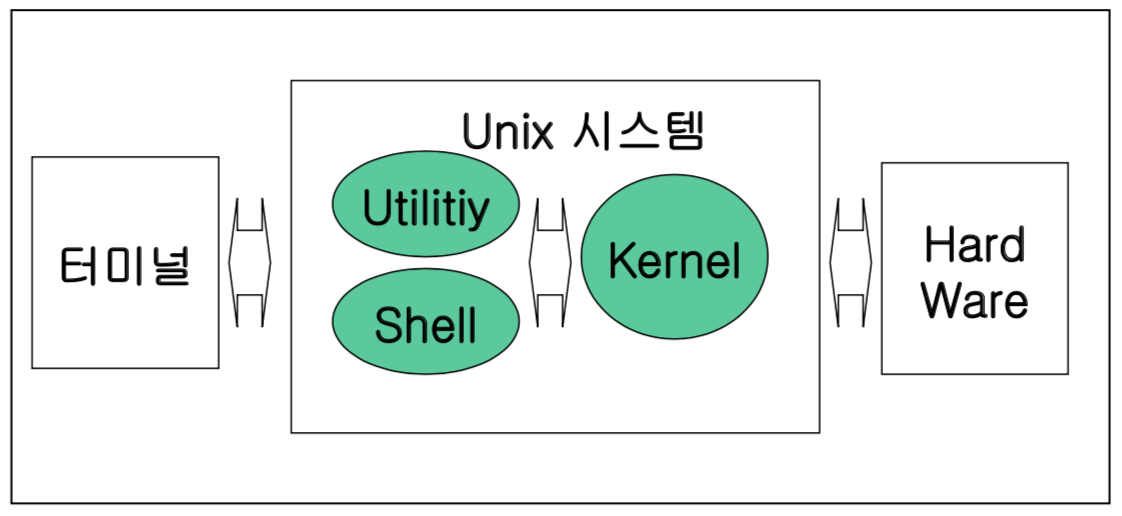
4) 대화형 시스템

5) 셸 프로그래밍

6) 계층적 파일 시스템

7) 다양한 SW 개발 도구

8. Unix의 구조



1) 커널(Kernel): 파일 관리, CPU 스케줄링, 메모리 관리, 입출력 서비스

2) 셸(Shell): 명령어 번역기, 커널과 사용자 사이의 인터페이스 제공

3) 유틸리티: 사용자의 편의를 위해 준비된 프로그램

9. Linux 탄생, 특징

1) Minix보다 좋은 OS를 만드는 것이 목표

2) 1991년 Linus Torvalds에 의해 최초 개발

3) 일반적으로는 커널, 컴파일러, 셸, 기타 응용프로그램이 조합된 배포판을 의미

4) 엄격히는 커널만을 의미

5) POSIX(Portable OS Interface for uniX) 지원: 어떤 기종에도 호환

6) UNIX와 유사

7) 임베디드, PC, 워크스테이션 등에 사용되는 인기있는 OS

8) 오픈소스 SW(GNU GPL)

- GPL 라이센스를 따르면 소스코드를 공개해야 함

- 상업적으로 사용 불가

10. Linux의 장점

1) 멀티 태스킹, 멀티 유저

2) 강력한 네트워킹 기능

3) UNIX 표준과 호환

4) 편리한 사용 환경 - CLI(셸), GUI(GNOME, KDE)

5) 멀티 플랫폼 지원 - LILO(LInux LOader), GRUB(Grand Unified Bootloader)

6) 다양한 파일 시스템 지원

7) 다양한 프로세스/쓰레드 간 통신 지원

8) 효율적인 HW 자원 관리

11. Linux의 단점

1) 문제 발생시 보상받을 수 없음

2) 보안 취약할 것이라는 선입관

3) 한글 입출력 어려움

4) IDE, 응용프로그램 부족

5) Linux는 오직 POSIX 기반 스케줄러만 가지고 있음

12. Kernel

1) OS와 사용자를 연결해주는 역할

2) 세부 기능

(1) 프로세스 관리

(2) 메모리 관리

(3) 파일 시스템 관리

(4) 장치 관리

(5) 네트워크 관리

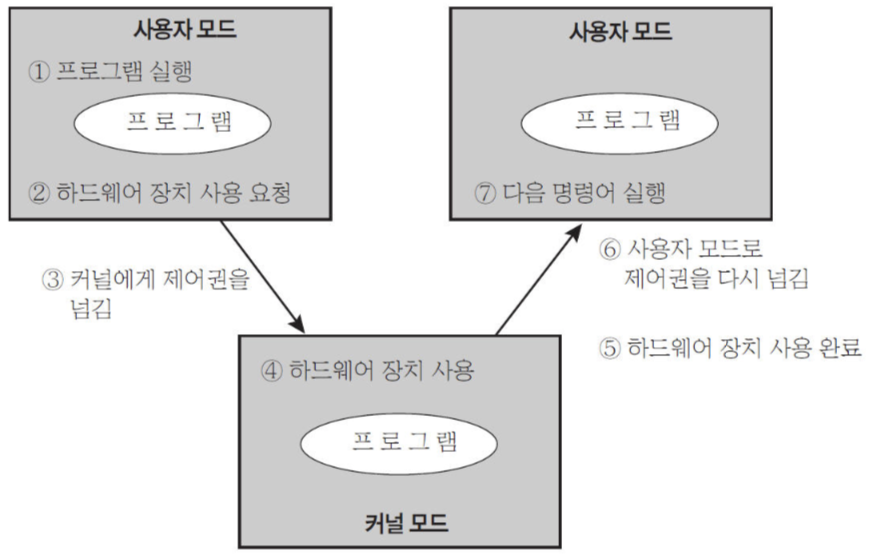
3) 구조

(그림)

Application <-> System Call Interface <-> HW Control <-> HW

4) 사용자 모드, 커널 모드

(그림)



5) 특징

(1) 모노리식(Monolithic) 커널: OS의 모든 기능이 하나의 주소공간에서 한 프로그램으로 수행되는 커널

(2) 커널 모듈 지원: 모노리식 커널의 크기, 성능 문제 해결

6) 버전

(1) 안정 버전(Stable Ver.): 두 번째 마이너 번호 짝수

(2) 개발 버전(Beta Ver., Development Ver.): 두 번째 마이너 번호 홀수

13. Linux 배포판 종류

1) 레드햇 페도라

2) 데비안

3) 젠투 리눅스

4) 우분투 리눅스

14. Linux 응용 분야

1) 임베디드 리눅스

2) 데스크탑 리눅스

3) 서버 리눅스: 서버 시장 50% 점유

4) 클러스터링, 슈퍼 컴퓨터: 가성비 우수

15. 디스크 파티션의 개념과 종류

- 파티션: 디스크 공간을 논리적으로 분할한 공간

1) 주 파티션: 물리적으로 분리된 디스크처럼 작동, OS 설치 가능

2) 확장 파티션: 주 파티션이 4개 이상 분할이 안되는 한계 극복

3) 논리 파티션: 개수 제한 없음, 단독으로 부팅 불가

16. Gnome, KDE 비교

17. GRUB, LILO 비교

1) LILO

(1) 과거 기본 부트로더

(2) 특정 파일 시스템에 의존하지 않음

2) GRUB

(1) 현재 기본 부트로더

(2) 동일한 시스템에 여러 OS 가질 수 있음

(3) OS, 네트워크 구성 편리

제2장 Virtual Machine

1. 가상화 기술(Virtualization)의 정의

- 물리적으로 다른 시스템을 논리적으로 통합하거나, 하나의 시스템을 논리적으로 분할해 활용하는 기술

2. 가상화 기술의 기대효과

1) 물리적 제한 극복

2) 자원을 효율적으로 사용

3) 유지보수 비용 절감

4) 복구 프로세스 개선

3. 가상화 유형

1) 서버: 플랫폼 기반, Hypervisor 기반

(1) 플랫폼 기반

가) CPU, Memory를 가상화하여 서비스 제공

나) 자원 간 모든 통신을 가상화

다) 독립성 확보(물리적인 서버 수 및 TCO(총 소유비용) 절감)

(2) Hypervisor 기반

가) 부트로더와 커널을 지닌 채 설치되어 작동

나) 가장 널리 보급된 가상화

다) 가상 서버와 HW 사이에 추상화 레이어 배치

라) OS 수정하지 않고 바로 설치 가능

2) 스토리지: RAID, SCSI 등

3) 백업

4) 애플리케이션: DBMS 가상화 등

4. 가상 머신의 이점

1) 호환성: 모든 표준 x86 컴퓨터와 호환

2) 격리: 마치 물리적으로 분리된 것처럼 분리

3) 캡슐화: 전체 컴퓨팅 환경을 캡슐화

4) 하드웨어 독립성: 기본 HW와 독립적으로 실행

5. 서버 가상화 기술의 종류

1) 전가상화: Hypervisor를 이용해 가상 서버들을 독립적으로 이용

2) 반가상화: 독립성 없이 상호작용

3) OS레벨 가상화: 하이퍼바이저 사용하지 않고, 가상화 기능을 호스트 OS에 포함

6. 스냅샷(Snapshot)

- 임의 시점에 가상 시스템의 정보와 상태를 저장해놓고 문제가 발생할 경우 되돌릴 수 있는 기능

제3장 CentOS7 Linux Installation

1. CentOS(Community ENTerprise OS)

- RHEL(RedHat Enterprise Linux)를 리빌드하여 무료 Enterprise급 플랫폼을 제공할 목적으로 개발된 Linux

2. GNU 프로젝트(GNU's Not Unix!)

- 1984년 리처드 스톨먼에 의해 시작

- FSF(Free Software Foundation, 자유 소프트웨어 재단)

- 누구나 SW를 자유롭게 사용하도록 하는 것이 목표

- 대부분 GPL(General Public License)

3. Fedora Project가 하는 일

1) 무료 및 오픈소스 SW 플랫폼 구축

2) 오픈소스 생태계 발전

4. CentOS에서 사용 가능한 웹 브라우저

1) Chrome(Chromium)

2) Firefox

3) Opera 등

5. 웹 브라우저의 플러그 인이란?

- 웹 브라우저의 일부로서 쉽게 설치되고 사용될 수 있는 프로그램

- 헬퍼(Helper)라고도 불림

- Adobe Flash, Java, Silverlight 등

6. YUM, Repository

0) RPM(Redhat Package Manager)

1) YUM(Yellow dog Updater, Modified)

- RPM의 패키지 의존성 문제 해결

- 인터넷을 통해 필요한 파일을 Repository에서 다운로드해서 설치하는 방식

2) Repository

- 설치 가능한 패키지 정보를 가지고 있는 저장소

제4장 CentOS Linux Command

1. 용어

1) X Window: MS의 Windows와 유사한 GUI 환경을 제공

2) GNOME(GNU Network Object Model Environment): 유닉스 계열 OS를 위한 GUI 테스크톱 환경

3) KDE(K Desktop Environment): GNOME에 비해 다양한 기능 제공 but 복잡하고 어려움

4) Terminal(Console): 유닉스와 유사한 TUI 환경을 제공

2. Terminal 환경에서 시스템 종료하기

1) poweroff

2) halt

+) halt -p

3) init 0

4) shutdown

(1) shutdown -P +10 : 10분 후 poweroff

(2) shutdown -r 22:00 : 22:00에 reboot

(3) shutdown -c : cancel

(4) shutdown -k +20 : 20분 후 현재 사용자들에게 종료 통보

3. Terminal 환경에서 재부팅하기

1) shutdown -r now

2) reboot

3) init 6

4. Runlevel

level 0: Power Off // poweroff.target

level 1: Single-User Mode(응급복구모드) // rescue.target

level 2: Multi-User(사용 안함) // multi-user.target

level 3: Multi-User(Text Mode) // multi-user.target

level 4: Not Used // multi-user.target

level 5: Multi-User(X Window Mode) // graphical.target

level 6: Reboot // reboot.target

# Runlevel 변경하는 실습

5. 가상콘솔

1) 가상의 모니터로 한 대에 컴퓨터에 모니터가 6대 연결된 효과

2) 총 6개의 가상 콘솔 제공

3) Ctrl+Alt + F1 = X-Window 터미널(1번째)

4) Ctrl+Alt + F2~F6 = Text Mode 터미널(2~6번째)

# useradd kim

# passwd kim

# who // 현재 로그인 된 사용자 확인

6. 명령어의 구조

[root@localhost /root]#\_

로그인 계정 + @ + 호스트 이름 + 현재 디렉토리 + 프롬프트 + 커서

7. hostname 변경

# hostnamectl status // hostnamectl: hostname control, status는 조회

# hostnamectl set-hostname gachon

# hostname

# reboot

8. 프롬프트

- 사용자의 명령 입력을 기다리는 표시

- 일반 사용자: $, 시스템 관리자: #

9. 명령 행 편집 방법

- 프롬프트 다음에 명령을 입력

- 문자 지우기: 백스페이스키 or Delete키

- 단어 지우기: Ctrl + W

- 문장 지우기: Ctrl + U

10. Linux 명령어의 구조

- 명령 [옵션] [인자...]

1) 명령: 리눅스를 사용하기 위한 수백 가지 명령

2) 옵션: 명령의 세부 기능 선택(-로 시작, 대소문자 구분)

3) 인자: 명령으로 전달되는 값

11. 기초 명령어

1) date: 현재 날짜와 시간 출력

2) clear: 현재 보이는 화면 지우고, 커서를 화면 좌측 상단으로 이동

3) man: 명령어의 사용법 출력

4) passwd: 비밀번호 변경

5) exit: 터미널 종료

6) Ctrl + D: 터미널 종료

12. 리눅스 파일의 종류와 특징

1) 일반 파일

- 데이터를 저장하는데 주로 사용

- 각종 텍스트 파일, 실행 파일, 이미지 파일 등 대부분의 파일

- 바이너리 파일이라고도 함

- 텍스트 이외에 파일들은 특정 응용프로그램이 있어야 확인 가능

2) 디렉터리

- Linux는 디렉터리도 파일로 취급

- 해당 디렉터리에 저장된 파일이나 하위 디렉터리에 대한 정보 저장

3) 심벌릭 링크

- 원본 파일을 대신하여 다른 이름으로 지정한 것

4) 장치 파일

- 하드디스크, 키보드 등 장치들도 파일로 취급

# file .bash\_profile // .bash\_profile의 file 종류 확인

13. 디렉터리

- Linux는 효율적인 관리를 위해 트리 구조로 디렉터리 구성

1) 작업 디렉터리

- 현재 작업중인 디렉터리(working directory)

- .으로 표시

- pwd 명령어로 확인(print working directory)

2) 홈 디렉터리

- 각 사용자에게 할당된 디렉터리

- ~로 표시

14. 절대 경로명과 상대 경로명

1) 절대 경로명

- 항상 루트(/)부터 시작

- 루트부터 해당 디렉터리까지 거치는 모든 중간 디렉터리 표시

- 절대 경로명은 항상 동일

2) 상대 경로명

- 현재 디렉터리 기준

- / 이외의 문자로 시작

- 상위 디렉터리는 ..

- 상대 경로명은 다를 수 있음

15. 파일과 디렉터리 이름 규칙

1) '/' 사용 X

2) 알파벳, 숫자, -, \_, .만 사용

3) 공백 문자, \*, ", ', % 등 사용 X

4) 대소문자 구별

5) .으로 시작하면 숨김 파일

16. cd

- 디렉터리 이동, change directory

- cd // (default) 해당 계정의 홈 디렉터리로 이동

- cd /home/user // 절대 경로명 사용

- cd 다운로드 // 상대 경로명 사용

- cd ~ // 홈 디렉터리로 이동

17. ls

- 디렉터리 내용 출력, list

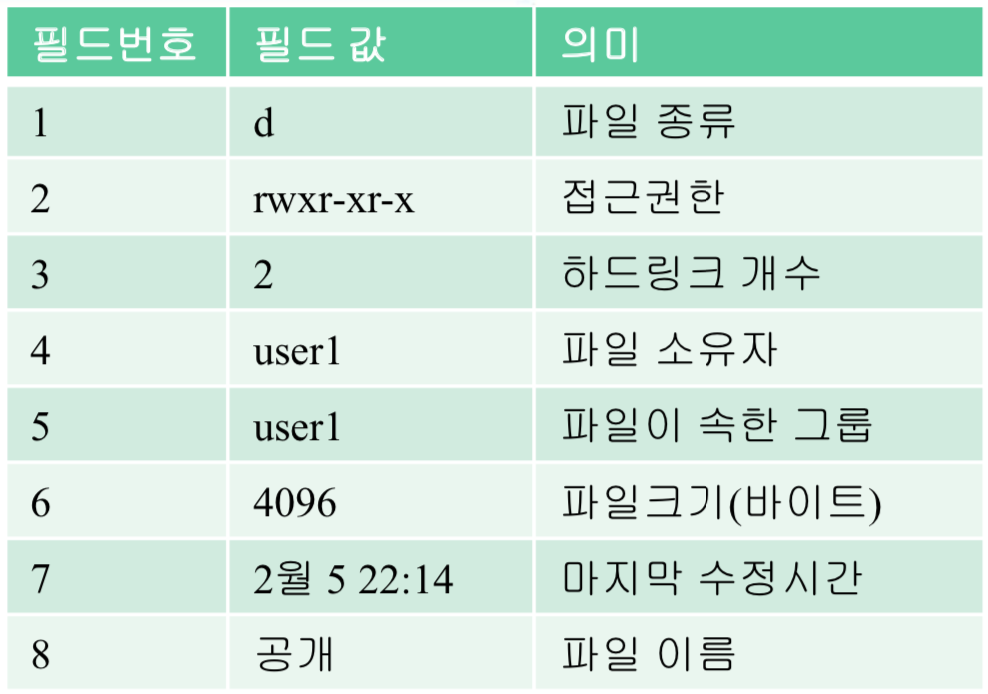
- ls -a // 숨김 파일 확인

- ls -F // 파일의 종류 표시

- 디렉터리(/), 심벌릭 링크(@), 실행 파일(\*), 일반 파일()

- ls /tmp // tmp의 디렉터리 내용 출력

- ls -l // 상세 정보 출력



- ls -d // 해당 디렉터리 자체 정보 출력

- ls -i // 해당 디렉터리에 속한 파일들의 inode 번호 출력

- ls -C // 디렉터리를 한 줄로 표시

- ls -R // 서브 디렉터리 내용 표시

- ls .bash\_profile // .bash\_profile이 있는지 출력

- dir, vdir: ls의 심벌릭 링크

18. mkdir

- 디렉터리 생성, make directory

- mkdir temp // 디렉터리 1개 생성

- mkdir tmp1 tmp2 tmp3 // 디렉터리 여러 개 생성

- mkdir -p tmp/mid/han // 중간에 디렉터리가 없을 경우 모두 생성

19. rmdir

- 디렉터리 삭제, remove directory

- rmdir tmp3 // 디렉터리 삭제

- 디렉터리 비어있지 않으면 삭제 불가(rm 명령어로 가능)

- rmdir -p tmp3 // 부모 디렉터리가 빈 디렉터리일 경우 자동으로 삭제

20. cat

- 파일 내용 출력, concatenate

- cat /etc/hosts // 파일 내용 출력

- cat -n /etc/hosts // 행 번호 붙이기

- cat > test.txt // 내용 입력 후 test.txt로 저장(덮어쓰기)

- cat < test.txt // test.txt의 내용을 출력

- cat >> test.txt // 내용 입력 후 test.txt에 append

- cat < test.txt > test1.cp // test.txt의 내용을 test1.cp로 복사

21. more

- 파일 내용 화면 단위 출력, more

- 다음 페이지(spacebar), 한 줄씩 스크롤(enter), 문자열 검색(/문자열), 종료(q)

- more +5 /etc/services // 5번째 행부터 출력

22. less

- 파일 내용 화면 단위 출력 + 스크롤 가능

- 다음/이전 행 스크롤(j, k), 다음 페이지(Ctrl+F, Spacebar), 이전 페이지(Ctrl+B)

23. tail

- 파일 뒷부분 출력(default값 10)

- tail +5 /etc/services // 5행부터 끝까지 출력

- tail -7 /etc/services // 마지막 7행 출력

- tail -f /etc/services // 주기적으로 반복 출력

24. cp

- 파일이나 디렉터리 복사, copy

1) cp (파일1) (파일2) // 파일1을 파일2의 이름으로 복사

2) cp (파일1) (디렉터리1) // 파일1을 디렉터리1로 복사

3) cp (파일1) (파일2) ... (디렉터리1) // 파일들을 디렉터리1로 복사

4) cp -i (파일1) (파일2) // 덮어쓸 것인지 물어봄

5) cp -r (디렉터리1) (디렉터리2) // 디렉터리 복사 시 반드시 지정

- 이미 존재하는 디렉터리면 그 아래 디렉터리로 복사

25. mv

- 파일이나 디렉터리 이동, 이름 바꾸기, move

1) mv (파일1) (파일2) // 파일1을 파일2로 이름 바꾸기

2) mv (파일1) (디렉터리1) // 파일1을 디렉터리1로 이동

3) mv (파일1) (디렉터리1/파일2) // 파일1을 디렉터리1로 이동, 파일2로 이름 바꾸기

4) mv (파일1) (파일2) ... (디렉터리1) // 파일들을 디렉터리1로 이동

5) mv -i (파일1) (파일2) // 덮어쓸 것인지 물어봄

6) mv (디렉터리1) (디렉터리2) // 디렉터리명 바꾸기

- 이미 존재하면 해당 디렉터리 아래로 이동

26. rm

- 파일이나 디렉터리 삭제, remove

1) rm (파일,디렉터리) // 삭제

2) rm -i data1 // 정말 삭제할 것인지 묻기

3) rm -r (디렉터리) // 디렉터리 삭제(내용 있어도 가능)

27. 파일 링크

1) 파일 링크: 파일에 새로운 파일명을 붙이는 것

2) 하드 링크: 기존 파일에 새로운 파일명을 붙이는 것

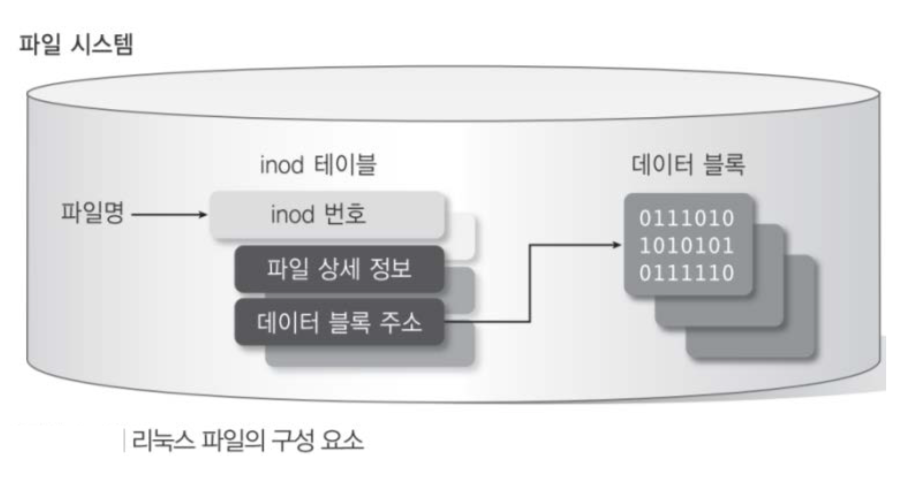
3) 심볼릭 링크: 원본 파일을 가리키는 새로운 파일 생성

28. Linux 파일의 구성

- 파일명 + inode + 데이터 블록 주소(-> 데이터 블록)

1) inode: 파일에 대한 정보(파일의 종류, 크기, 소유자 등, ls -l)

- inode 번호 확인은 ls -i



29. ln

- 파일 링크 만들기, link

- ln test test.ln // lntest라는 test의 하드 링크 생성(inode 같음)

- ln -s test test.sl // lntest라는 test의 심볼릭 링크 생성(inode 다름)

30. 심볼릭 링크의 특징

1) 파일의 종류 l로 표시

2) 하드 링크 개수 1개

3) 파일명 뒤에 원본 파일명 test.sl -> test

4) inode 번호 다름(별개의 파일)

5) 디렉터리에도 생성 가능

6) 파일 시스템이 달라도 생성 가능

31. touch

- 빈 파일 생성 or 수정 시간 변경

1) touch test // test 파일 생성(없을 경우)

2) touch test // 수정 시간을 현재 시간으로 변경(있을 경우)

3) touch -t 01011200 test // 1월1일 12:00로 변경

32. grep

- 파일의 내용을 검색

1) grep DHCP data // data파일에서 DHCP 문자열이 포함된 행 검색

2) grep -n DHCP data // 행 번호 같이 출력

33. find

- 특정 조건으로 파일 자체를 검색

1) find /usr -name ls // usr 디렉터리에서 ls라는 파일명으로 검색

2) find /home -user kim // home 디렉터리에서 kim 유저가 소유한 파일 검색

3) find /tmp -user kim -exec rm {} \;

- tmp 디렉터리에서 kim 유저가 소유한 파일을 찾아서 삭제

- {}는 검색된 파라미터들을 전달하는 곳

- 하나씩 확인하려면 -ok 명령 사용

34. whereis

- 정해진 디렉터리의 명령어 위치 검색

- whereis mv // mv 명령어의 위치 검색

35. which

- 에일리어스나 PATH 환경 변수에서 명령어 위치 검색

- which mv // mv 명령어의 위치 검색

36. du

- 디렉터리 용량 확인, directory usage

- du -h

37. df

- 디스크 파티션 사용량 확인, disk free

38. chmod

- 파일, 디렉터리의 허가권 변경, change mode

- chmod 770 install.log

- 퍼미션을 770로 변경

- 7(4+2+1) 7(4+2+1) 0(0+0+0), rwx\_rwx\_---(소유자, 그룹, 그 외 사용자 순서)

39. chown

- 파일의 소유자 변경, change owner

- chown park test // test 파일의 소유자를 park으로 변경

제5장 VI Editor

1. Linux 편집기의 종류

1) 행 단위 편집기

(1) ed: 유닉스 초기 행 편집기(거의 사용 X)

(2) ex: vi에 연결하여 vi를 강력하게 하는 기능 제공

(3) sed: 스트림 편집기, 명령을 일괄적으로 바꿔서 출력

2) 화면 단위 편집기

(1) vi: Linux에서 일반적으로 사용하는 편집기

(2) emacs: 기능 다양하지만 복잡하고 전문적

3) GUI 편집기

(1) gedit: GUI 텍스트 편집기

2. 모드형, 비모드형 편집기

1) 모드형

- 입력 모드, 명령 모드 구분

(1) 입력 모드: 텍스트를 입력하는 모드

(2) 명령 모드: 수정, 삭제, 복사, 붙히기 등 편집 모드

- vi는 모드형에 해당

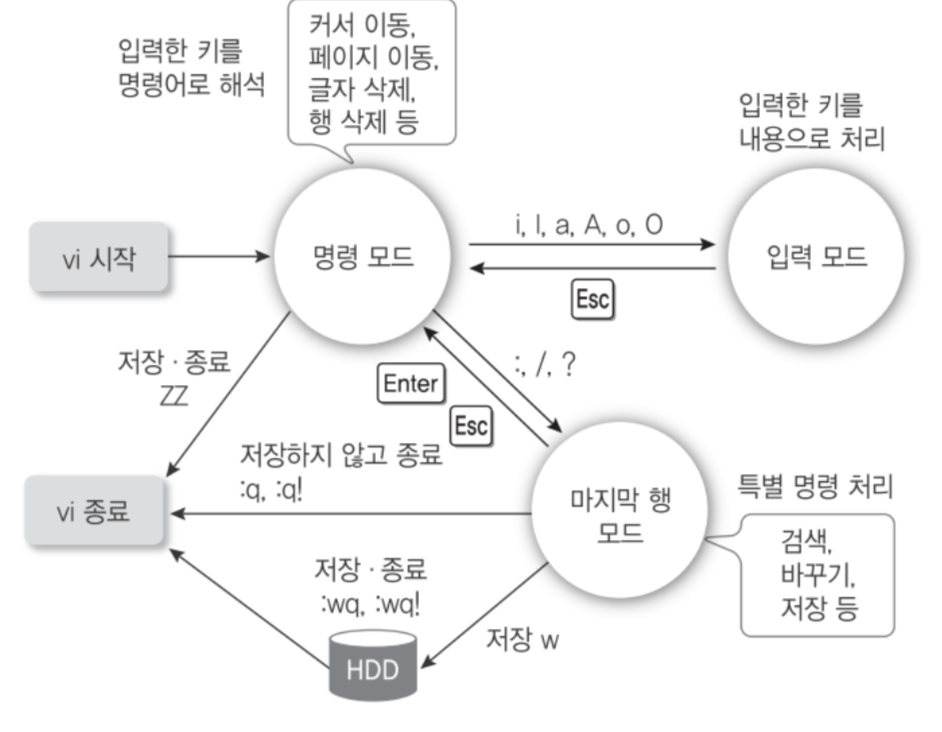
2) 비모드형

- 입력 모드, 명령 모드 구분 X

- Ctrl, Alt와 같은 특수 키 사용

- 한글, Word가 비모드형에 해당

3. vi 동작 모드를 그림을 그려 설명하시오.



4. vi 시작

1) vi test.txt // vi 시작(없으면 생성)

2) vi // 빈 파일로 시작

5. vi 저장 및 종료

1) 마지막 행 모드

(1) :q // 작업한 내용이 없을 때 그냥 종료(quit)

(2) :q! // 저장하지 않고 강제 종료

(3) :w [파일명] // 파일명으로 저장 (종료 X)(write)

(4) :wq, :wq! // 저장하고 vi 종료

2) 명령 모드

(1) ZZ // 저장하고 종료

6. 입력 모드로 전환하기

1) i - 커서 앞에 입력

2) a - 커서 뒤에 입력

3) o - 커서 다음 행에 입력

4) I - 커서가 위치한 행의 첫 칼럼에 입력

5) A - 커서가 위치한 행의 마지막 칼럼에 입력

6) O - 커서가 위치한 행의 앞 행에 입력

7. 커서 이동 키

1) k, j - 위, 아래

2) h, l - 왼쪽, 오른쪽

- 방향키 사용 가능

3) ^, O - Home키

4) $ - End키

8. 화면 이동 키

1) Ctrl + U/D - 반 화면 위/아래

2) Ctrl + B/F - 한 화면 위/아래

3) Ctrl + Y/E - 한 행 위/아래

4) G - 마지막 행으로 이동(명령 모드)

5) 10 + G - 10행으로 이동(명령 모드)

6) :20 + Enter - 20행으로 이동(마지막 행 모드)

7) :$ - 마지막 행으로 이동(마지막 행 모드)

9. 내용 수정

1) r - 한 글자 수정

2) cw, #s - 한 단어 수정

3) C - 해당 커서가 지워지고 입력 모드로 전환

4) cc - 현재 행 모두 지워지고 입력 모드로 전환

10. 내용 삭제

1) x, #x - 커서 위치 글자 삭제

2) dw, #dw - 커서 위치 단어 삭제

3) dd, #dd - 커서 위치 행 삭제(잘라내기)

4) D - 커서 위치부터 행의 끝까지 삭제

11. 명령 취소

1) u - 명령 취소

2) U - 해당 행에서 한 모든 명령 취소

3) :e! - 마지막 저장한 내용으로 복귀

12. 복사하기, 잘라내기

1) yy, #yy - 커서가 위치한 행 복사

2) p - 커서가 위치한 행 아래쪽에 붙여넣기

3) P - 커서가 위치한 행 위쪽에 붙여넣기

4) dd, #dd - 행 잘라내기

13. 언네임드 버퍼, 네임드 버퍼

1) 언네임드 버퍼: 이름을 붙이지 않은 버퍼

2) 네임드 버퍼: 이름을 붙여서 사용하는 버퍼

- "a, "b, "1, "2 등 사용

- "a + yy - "a 버퍼에 복사한 내용 저장

- "a + p - "a 버퍼의 내용을 커서 아래 행에 붙여넣기

14. 마지막 행 모드에서 범위 지정 및 작업

1) 범위 지정

(1) :10, 20 - 10행부터 20행

(2) :1, $ - 1행부터 마지막 행

(3) :1, . - 1행부터 커서가 있는 행

(4) :., $ - 커서부터 마지막 행

(5) :, -3 - 현재 행과 이전 3행까지 지정(총 4행)

2) :#y, :#d - #행만큼 복사, 잘라내기

3) :<범위>y, :<범위>d - <범위>만큼 복사, 잘라내기

4) :pu - 현재 행 다음에 버퍼 내용 붙여 넣기

15. 검색하기

- :가 아닌 /나 ?로 마지막 행 모드 진입

1) /문자열 - 아래 방향으로 검색

2) ?문자열 - 위 방향으로 검색

3) n - 다음 검색

4) N - 이전 검색

16. 바꾸기

1) :s/문자열1/문자열2/ - 첫 번째로 나오는 문자열1을 문자열2로 치환

2) :%s/문자열1/문자열2/g - 파일 전체에서 치환

3) :<범위>s/문자열1/문자열2/ - 범위에서 치환

4) :<범위>s/문자열1/문자열2/gc - 범위 내 모든 문자열 치환하고 수정 여부 묻기

17. 기타 명령어

1) :r 파일명 - 지정한 파일의 내용을 현재 커서에 삽입

2) :e 파일명 - 파일 전환(:w 이후에 실행해야 함)

3) :n - 여러 파일을 실행했을 경우 다음 파일로 전환

4) :! 셸 명령 - 셸 명령 한 줄 실행 후 Enter키로 vi 복귀

5) :sh - 셸 명령 여러 줄 실행 후 exit 명령어로 vi 복귀

6) Ctrl + l - 현재 화면 다시 출력

7) Ctrl + g - 현재 행 번호 마지막 행에 출력

8) J - 현재 행과 그 아래 행을 연결하여 한 행으로 만듦

9) . - 바로 직전에 실행한 명령 반복

18. vi 환경 설정 방법 3가지

1) 사용자 홈 디렉터리에 .exrc 파일로 저장

- vi의 환경 설정 파일, 사용자 또는 디렉터리마다 다르게 설정 가능

- set nu

set list // 이 내용을 .exrc로 홈 디렉터리에 저장하면 된다.

2) 환경 변수 EXINIT에 저장

- 셸의 설정 상태를 저장하는 환경 변수

- EXINIT='set nu list'

export EXINIT

3) vi의 마지막 행 모드에서 명령으로 실행(vi 종료하면 해제됨)

19. vi 환경 설정 명령

1) set nu, set nonu - 행 번호 표시, 숨기기

2) set list, set nolist - 특수 문자 표시, 숨기기

3) set, set all - 환경 설정 상태 표시, 전체 표시

제6장 Shell

1. 셸의 기능 3가지

1) 명령어 해석기: 사용자와 커널 사이의 명령어 전달

2) 프로그래밍 기능

3) 사용자 환경 설정 기능

2. 셸의 종류

1) 본 셸(Bourne Shell)

- Unix 최초의 셸

- 처리 속도가 빨라 많이 사용되었고, 지금도 많은 셸의 기반

- 편의 기능 부족

- 명령 이름 sh

- 현재 본 셸은 bash로 심볼릭 링크 되어있음

2) C 셸(C Shell)

- 빌 조이(Bill Joy) 개발

- 에일리어스, 히스토리 등 편의 기능 포함

- 구문 형식이 C언어와 같음

- 명령 이름 csh, 기본 프롬프트 얘만 %

3) 콘 셸(Korn Shell)

- C 셸과 달리 본 셸과의 호환성 유지

- 에일리어스, 히스토리 등 편의 기능 제공하면서 처리 속도 빠름

- 명령 이름 ksh

4) 배시 셸(Bash Shell)

- 본 셸과 호환성 유지하면서, C 셸, 콘 셸 장점 흡수

- 명령 이름 bash

- GPL 라이센스, Linux의 기본 셸

+) grep kim /etc/passwd로 기본 셸 확인 가능

3. chsh

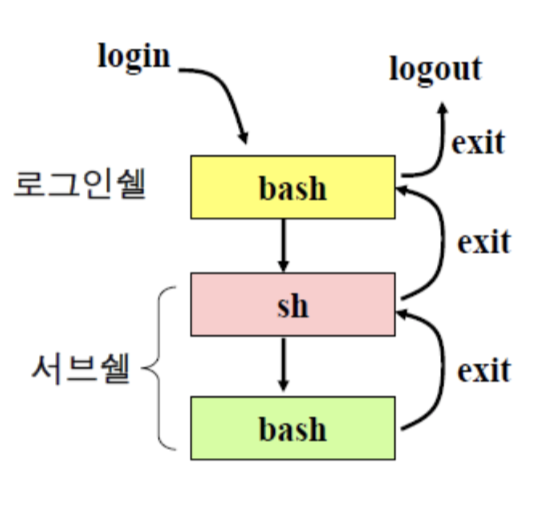
- 기본 셸 바꾸기, change shell

- cat /etc/shells : 바꿀 수 있는 shell의 종류 출력

- echo $SHELL : 현재 사용중인 셸 확인

- chsh -s /bin/csh kim : kim의 기본 셸을 C 셸로 변경(절대경로 사용해야함)

4. 로그인 셸, 서브 셸



1) 로그인 셸: 로그인 직후 자동으로 생성되는 셸

2) 서브 셸: 사용자가 직접 생성한 셸

3) 서브 셸 종료는 Ctrl + D, exit 명령어 사용

4) 셸 중첩해서 생성 가능, 종료하면 이전 셸로 복귀

5) 로그인 셸에서 로그아웃하면 접속 해제

5. Shell 출력하기

1) echo : 한 줄 출력

- echo linux // linux

- echo "linux fedora" // linux fedora

2) printf : 형식 지정하여 출력

- printf "text\n"

- printf "%d + %d = %d\n" 10 10 20 // 10 + 10 = 20

6. Shell 특수문자

1) \* : 0개 이상의 문자(와일드카드)

2) ? : 하나의 문자(와일드카드) / [] : 하나의 문자(정규식)

3) ~ : 현재 사용자의 홈 디렉터리 (~user2/는 user2의 홈 디렉터리)

4) - : cd 이전에 작업하던 디렉터리

5) ; : 명령어 순차적으로 실행

6) | : 왼쪽의 명령어의 결과를 오른쪽으로 전달하여 실행

7) '' : 모든 특수 문자의 기능을 없애고 문자열로 감싼다

8) "" : $, ₩, \을 제외한 특수 문자의 기능을 없애고 문자열로 감싼다

9) ‛‛ : 감싼 문자열을 명령어로 해석하여 실행

10) \, ₩ : 바로 뒤에 나오는 특수 문자의 효과를 없앰

11) >, <, >> : 입출력 방향 전환

7. 파일 디스크럽터

1) 파일 관리를 위해 붙이는 일련 번호

- 0(stdin): 표준 입력

- 1(stdout): 표준 출력

- 2(stderr): 표준 오류

2) 입출력 장치를 변경할 때는 파일 디스크럽터를 이용

3) 리다이렉션: 표준 입출력 장치를 파일로 바꾸는 것

8. 출력 리다이렉션(>, 1>)

1) > : 파일 덮어쓰기

- ls -al > out1 : 명령의 결과를 out1 파일에 저장

- ls -al 1> out1과 동일

2) >> : 파일에 내용 추가

- date >> out1 : 명령의 결과를 out1 뒤에 추가

9. 오류 리다이렉션(2>)

- 출력 리다이렉션에서는 오류 내용은 전달하지 않음

- ls /abc 2> ls.err : /abc가 없다는 오류 내용을 ls.err에 저장

- ls /abc 2> /dev/null : 오류 메시지 버리기

10. 입력 리다이렉션(<, 0<)

- cat out1, cat < out1, cat 0< out1 : 모두 동일한 결과

11. 셸 변수와 환경 변수

1) 셸 변수(지역 변수): 현재 셸에서만 사용 가능, 서브 셸 X

2) 환경 변수(전역 변수): 서브 셸로도 전달

12. 변수 출력

1) set : 셸 변수와 환경 변수 모두 출력

2) env : 환경 변수만 출력

3) echo $SHELL : 특정 변수 출력

13. 변수 설정하기

1) 셸 변수 설정하기

SOME=test

echo $SOME // test 출력

- 변수 이름과 문자열 사이에 공백 있으면 안됨

2) 환경 변수 설정하기

export SOME // 설정된 셸 변수를 환경 변수로 설정

export SOME1=test1 // 동시에도 가능

3) 환경 변수 취소

export -n SOME1

4) 변수 취소

unset SOME

14. 에일리어스

- 기존의 명령을 대신하는 별명, alias

- 긴 명령을 짧게 만들 수 있음

- 여러 명령을 연결하여 하나로 만들 수 있음

- alias // 기존 alias 확인

- alias rm='rm -i' // rm -i 명령에 rm이라는 alias 부여

- bash 셸에서는 인자 전달 불가 -> 프로그래밍 기능 사용

function cdpwd {

cd $1; pwd

}

cdpwd /tmp

- unalias rm // 에일리어스 해제

15. 히스토리

- 사용자가 이전에 입력한 명령을 다시 불러 사용하는 것, history

- history // 이전 명령 보기

- !! : 직전 명령 재실행

- !번호 : 히스토리에서 해당 번호 재실행

- !문자열 : 히스토리에서 해당 문자열로 시작하는 마지막 명령 재실행

- 오타났을 경우 윗 방향키 -> 좌우 방향키로 이동 후 수정하여 재실행하면 됨

- 로그아웃할 때 .bash\_history 파일에 히스토리를 저장한다.

16. 프롬프트 설정

- 프롬프트 설정 변수: PS1

- echo $PS1 // 현재 프롬프트 출력

- 이스케이프 문자 사용 가능

- PS1='`uname -n`' // 호스트 이름으로 프롬프트 설정

17. 환경 설정 파일

- 사용자가 로그인 할 때마다 자동으로 실행되는 명령을 저장한 파일

1) 시스템 환경 설정 파일

- 시스템을 사용하는 전체 사용자의 공통 환경 설정 파일

(1) /etc/profile : 공통으로 적용되는 환경 변수 설정

(2) /etc/bashrc : 공통으로 적용되는 함수, 에일리어스, 프롬프트 등

(3) /etc/profile.d/\*.sh : 언어, 명령 별 설정

2) 사용자 환경 설정 파일

- 각 사용자에 해당하는 설정 파일(숨김 파일)

(1) ~/.bash\_profile

(2) ~/.bashrc : 사용자가 정의한 함수, 에일리어스

(3) ~/.bash\_logout : 로그아웃 시 실행할 함수

-------------------

1. Shell Scripts Programming

\* 새로운 사용자를 등록하면 /etc/skel/ 디렉토리에 기본값으로 저장되어있는 파일들을 사용자의 기본 홈 디렉토리로 복사하여 생성

- 응용 프로그램을 배포하는데에도 많이 사용

2. 주요 내장 명령어

1) :(콜론) : null 명령어 (0 값을 돌려줌)

2) .(dot) : 현재의 셸 환경을 가지고 실행

3) break, continue

4) echo, exec, exit, export

5) read : 명령행에서 인수를 입력받음

6) return

3. 셸 스크립트(.sh) 작성 후 실행

1) vi name.sh

2) chmod 755 name.sh // 모든 사용자가 실행 가능하도록

4. 변수

1) 모든 변수가 string으로 취급

2) 대소문자 구분

3) =에 좌우공백이 없어야 함

4) $ 문자가 들어간 글자를 출력하려면 '' 사용하거나 앞에 ₩ 붙임 (""는 효과 없음)

5) exit 0 마지막에 붙여주기

5. 연산

num1=100

num2=`expr $num1 + 30` // 역따옴표 사용해야 함

// = 좌우엔 공백 없어야하고, 숫자 좌우엔 공백 있어야 함

// 연산하려면 expr 키워드 붙여줘야하고 \*는 ₩\*로 사용해야 함

6. 파라미터

파라미터는 $0, $1, $2... 로 사용, 전체 파라미터는 $\*로 표시

$0는 파일명

#!/bin/sh

echo "실행할 파일명은 <$0>이다"

echo "첫번째 파라미터는 <$1>..."

exit 0

7. if문

#!/bin/bash

if [ 조건 ] // [] 앞뒤에 공백 필수

then

echo "true statement"

else

echo "false statement"

fi

exit 0

8. 연산자

"string1"="string2"

-n "string" // null이 아니면 true

-z "string" // null이면 true

수식1 -eq 수식2 // 수식1=수식2

-ne (not equal)

-gt (greater than)

-ge (greater than equal)

-lt (less than)

-le (less than equal)

수식! // !

#!/bin/bash

if [ 100 -eq 200 ]

then

echo "100과 200은 같다"

else

echo "100과 200은 다르다"

fi

exit 0

-d 파일명 : 디렉토리면 true

-e 파일명 : (exist) 존재하면 true

-f 파일명 : 일반 파일이면 true

9. case문

#!/bin/bash

case "$1" in

start)

echo "시작!";; // break 대신 ;; 사용

stop)

echo "중지!";;

restart)

echo "재시작!";;

\*)

echo "기타!";;

esac

exit 0

#!/bin/bash

echo "리눅스 재미있나요? (Y/N)"

read answer

case $answer in

yes | y | Y | Yes | YES)

echo "좋아요"

echo "아주";;

[nN]\*)

echo "안좋아요";;

\*)

echo "잘못입력!"

exit 1;;

esac

exit 0

10. 관계연산자(AND, OR)

and = -a, &&

or = -o, ||

#!/bin/bash

echo "파일명 입력"

read fname

if [ ₩( -f ₩$fname ₩) -a ₩( -s ₩$fname ₩) ]

then

head -5 $fname

else

echo "파일이 없거나 0입니다!"

fi

exit 0

11. for문

#!/bin/bash

sum=0

for i in 1 2 3 4 5 6 7 8 9 10

do

sum=`expr $sum + $i`

done

echo "1부터 10까지의 합 : " $sum

exit 0

for fname in $(ls \*.sh)

12. while문

#!/bin/bash

while [1]

do

echo "Hello World"

done

exit 0

// [1] [:] [true] 는 무한루프

// Ctrl + C로 루프 탈출

#!/bin/bash

sum=0

i=1

while [ $i -le 10 ]

do

sum=`expr $sum + $i`

i=`expr $i + 1`

done

echo "sum은 " $sum

exit 0

#!/bin/bash

echo "비밀번호를 입력하세요!"

read mypass

while [ $mypass != "1234" ]

do

echo "틀립니다 다시 입력!"

read mypass

done

echo "맞습니다"

exit 0

13. until문

- while은 참일 동안 반복

- until은 거짓일 동안 반복(문법 같음)

14. break, continue, exit문

15. 사용자 정의 함수

#!/bin/bash

myFunction () {

echo "함수 안으로 들어옴"

return

}

myFunction // 호출

exit 0

16. eval

문자열을 명령문으로 인식하고 실행

#!/bin/bash

str="date"

echo $str

eval $str

exit 0

17. export

export var2="외부변수"

18. printf

#!/bin/bash

var1=100.5

var2="가천대 컴공"

printf "%5.2f \n \n %s" var1 "var2" // 문자열 공백때문에 "" 사용

exit 0

19. set, $(명령어)

Linux 명령어의 결과를 사용하고자 할 땐 $(date) 처럼 사용

echo "오늘 날짜는 $(date)입니다."

그 결과를 파라미터를 사용하고자하면 set 명령어 사용

set $(date)

echo "오늘은 $4 요일입니다."

.sh 파일 만들고 실행 속성

chmod 775 test.sh <<< 잊지말기!!