소프트웨어 프로젝트 1

(리눅스의 기초)

1. Linux란?

- Windows나 Mac OS 등과 같은 *운영체제* 이다.
(운영체제 [Operation System] = 응용 프로그램 및 사용자가 컴퓨터를 구성하는 장치들을 사용할 수 있도록 지원하는 소프트웨어)
- *OS 친구들* : Windows, Mac OS, Android, IOS, ...

2. Linus Torvalds

- 오픈 소스 형식으로 개발시작 -〉 공동 개발 프로젝트로 진행중
- 1991년에 Linux 0.0.1 Version 을 usenet에 처음으로 공개
- kernel.org를 통한 개발자 그룹을 통해 발전시킴

3. Linux Developer

- Linux Torvalds가 취미로 시작
- 개발 또는 지속적인 지원에 대한 전적인 책임을 특정인 또는 단체가 지지 않음.
- 1,000 명이 넘는 개발자가 매번 커널 릴리스가 있을 때 기여함

4. Linux가 사용되는 곳

- Server
- Desktop
- 세톱박스
- TV
- Digital Camera
- Navigation
- 블랙박스
- Smart Phone

5. Linux 가족들

Debian / Redhat 계열
Debian : Debian, Ubuntu, Xubuntu, Kubuntu, ...
RedHat : Redhat, Centos, ...

6. Linux

배포판 = Kernel + Library + Application
- Kernel : 작지만, 매우 중요하고 강력한 요소, 함부로 수정 할 수 없음.

7. Linux 배포판의 응용프로그램

웹브라우저, 썬더버드, 이메일, ftp, telnet, Torrent, 미디어 플레이어, 레코더, 편집기, Libre Office, Gimp(그래픽도구), 개발도구, 시스템 관리 도구, 서버 프로그램, ...

8. Linux History

- PPT 참고

9. Linux Kernel

Computer system의 모든 자원들을 관리함
(CPU, Memory, Hdd, I/O, Task, Process, File, ...)

10. CPU 관리

- 리눅스 커널안의 스케쥴러가 CPU를 어떻게 사용할 지 관리
  * 스케쥴러
  - CPU가 수행하는 작업 = CPU들의 계산 능력을 각 프로세스들에 할당
  - CPU가 해야 하는 작업이 없을 때 = 유휴상태 -〉 Sleep 모드 (Memory를 제외하고 모든 전력을 끔)
  - Sleep -〉 최대 절전 모드 (Memory 내용을 HDD로 옮기고, 모든 전원을 내린다.)
  - Memory 는 전원이 들어오지 않으면, 메모리의 내용이 모두 증발해버리기 때문

11. Memory 관리

- 커널은 코드나 데이터를 메모리에 저장 할 수 있다.
- 가상 메모리와 MMU -〉 무제한, 선형적, 독점적으로 메모리를 사용 할 수 있음

12. I/O Device (입출력 장치)

- 저장장치
  - HDD, ODD, Tape
- Network 장치
  - Ethernet, WI-FI

13. Process

- CPU, Memory를 사용하는 실행 중인 작업
- 프로세스 = 보통 우리가 프로그램이라고 말하는 것
- 사용자 프로세스, 시스템 프로세스

14. File

- DataStream
- FS (File System)이 관리함
- Linux 는 Device 도 File로 관리람 (/dev/~~)

15. User (사용자)

- Linux user
  - 리눅스를 사용하여 원하는 일을 수행하는 주체
  - 사용자 = 진짜 사람 or 서비스(프로그램)
  - 사용자는 어떤 그룹에 속할 수 있음
- 시스템에 대한 사용자권한
  - 파일 / 디렉토리에 대한 접근 권한은 사용자 또는 그룹별로 설정할 수 있음
- Multi User System
  - 한 시스템을 여러 사용자가 동시에 사용할 수 있음.

16. Kernel이 관리하는 자원

- 보안
- 라이브러리
- 소프트웨어
- 전력
- 커널 자체 (모니터링)

17. Kernel가 제공하는 기능

- MultiTasking
- MultiUser
- 공유 라이브러리
- 표준 네트워크 프로토콜
- Virual Memeory
- File System

18. Kernel = ?

- System Call (커널 함수 호출)
  - Apllication에서의 호출
  - Library에서의 호출
- 응용프로그램 -〉 하드웨어 (불가능)
- 응용프로그램 -〉 (라이브러리) -〉 Kernel -〉 하드웨어 (가능)

19. Linux 개발 환경 기초

- 사용자 로그인
  - Root User (Super User) = root = 0: 최고 관리자 (시스템 설정까지도 변경 가능)
  - 일반 유저 : 일정 한도 내에서 접근 권한이 있음
- Linux 사용방법
  - Shell에서 명령어를 입력하여 사용

20. Process

- 사용자가 요청하는 일을 수행하기 위해서 리눅스는 프로세스를 생성함
- 생성된 프로세스는 리눅스가 처리하는 일의 단위
- 사용자 프로세스 / 시스템 프로세스 / 데몬 프로세스 (=뒷쪽에서 이것저것 처리하는 녀석) / ...
- 일단은 프로세스 = 프로그램이라고 생각하자

21. File System

- 파일을 열고 쓰고 닫고 하는 일을 요청
- 파일 시스템은 파일과 디렉토리로 데이터를 관리함
- Linux File System 에서 지원하는 명령어들을 통해 해당 파일과 디렉토리와 관련된 일을 수행할 수 있음.

Shell 이란?
bash, sh 등과 같이 그냥 니들이 검은 색 창에서 명령어를 치면 받아서 응답해주는 녀석임

man [명령어]
[명령어]에 대한 메뉴얼을 보여줘라

cat [파일명]
[파일명]을 읽어서 내보내라

ls
-1 = 한줄씩
-a = 모든 것을 (숨김 폴더 까지)
-l = 세세하게 모든 정보를

pwd
지금 활동중인 디렉토리의 절대경로를 내놓습니다.

ㅋㅋㅋㅋㅋ
. = 현재
.. = 부모

cd
'..' = 부모
'dir' = 상대경로
'/bin' = 절대경로
'-' = 이전 [특수 디렉토리]
' ' = 홈(~) [특수 디렉토리]
'~' = 홈(~) [특수 디렉토리]

mkdir
디렉토리 만들기

rmdir
디렉토리 지우기

cp, mv
"이미 존재하면 덮어 씀"
-a = 원본 그대로 복사
-b = 기존의 각 대상 파일을 백업
-i = 덮어 쓰기 전에 확인
-l = 복사하는 대신 링크파일 생성
-r = 자식놈들까지 모두 다

rm
-f = 의견 없이 삭제
-i = 매번 지울때마다 사용자의 의견을 물음
-r = 디렉토리 내부에 있는 모든 정보들 까지 모두 삭제
-v = 삭제 결과들을 출력하면서 수행

패턴
~~₩~~* = 임의의 **문자열** 을 의미, 복수/null 까지 모두
? = 임의의 단일 **문자**
[a-s] = a ~ s까지 단일 문자

cat
나열된 파일들을 합치고 화면에 한번에 출력함

more
cat 처럼 읽어, 그리고 출력해!
대신 너는 스크롤리 구현되어있어야해!

head
cat 처럼 읽어, 대신 처음부터 일부만 보여줘

tail
cat 처럼 읽어, 대신 마지막부터 일부만 보여줘

tar
파일 여러개를 하나로 합치는 명령어.
합치는 것 : tar -cvf [파일명.tar] [폴더명]
푸는 것 : tar -xvf [파일명.tar]

bzip2, gzip
압축기능까지 있음

df
파일시스템에 남아있는 공간을 보여줌

du
각각의 파일이 차지하고 있는 용량을 보여줌

diff a.txt b.txt
a.txt와 b.txt에서 다른 것을 보여줌

sudo
- substitute user do 의 약자
- 관리자 권한 (root의 권한)으로 명령을 하기 위해 씀

id
- 내가 무슨 계정인지 알려줌

root
id = 0
Super User (Windows의 windows Administrator)
/etc/sudoers 에서 권한을 설정해주어야함

chown
파일과 디렉토리의 소유자를 설저함
-R = 하위 디렉토리까지 전부다

chmod
파일과 디렉토리의 사용권한 설정
-R = 하위 디렉토리까지 전부다

권한
M = 파일, 디렉토리, 장치, ...
R = 읽을 수 있는가
W = 쓸 수 있는가
X = 실행 할 수 있는가

PIPE and Redirection
표준입력 = 키보드,
표준출력 = 터미널에 나오는 글자

Pipe (|) = 표준 출력을 다른 명령의 표준 입력으로 연결시킴

ex) ls|more
ls에서 출력한 목록들이 more로 들어가서, more에서 처리해서 터미널에서 보여줌

ex) ls|cat
ls에서 출력한 목록들이 cat으로 들어가서, more에서 처리해서 터미널에서 보여줌
= 그냥 ls와 같음

Redirection (〉) = 출력 결과를 파일로 내보냄

ex) ls /bin 〉 ~/a.txt
/bin의 파일목록들을 ~/a.txt로 내보냄
ex) cat a.txt b.txt 〉 c.txt
a.txt 와 b.txt의 내용을 합쳐서 c.txt로 내보냄
ex) echo "hihihihihihasdasdasdasd" 〉 a.txt
hihihihihihasdasdasdasd 이란 글자를 a.txt로 내보냄
(echo명령어 = 재출력 명령어)

grep 〈문자열〉 〈파일〉
grep -n " the " correct.txt

wc
라인 개수
-l = 줄

텍스트 편집 기초 (vi, vim)
터미널 기반 텍스트 편집기
마우스나 방향키 없이도 사용가능
생산성을 향상시키기 위해 다른 도구들과 연동

vi 〈파일명〉
이미 있는 경우, 열고 없는 경우 그냥 빈파일로 함
아무것도 안넣으면 빈파일로 함

vi의 세가지 모드

명령어 모드
vi에서 제공하는 명령어들을 사용하는 모드
입력 모드
Insert, i, a

i = 현재 커서 앞쪽
I = 현재 줄의 맨 앞
a = 현재 커서 뒷쪽
A = 현재 줄의 맨 끝
o = 현재 커서가 있던 아랫줄부터 하나가 열려서 텍스트 입력 가능
O = 한줄 밀고

ex 모드
콜론을 이용해서 저장/종료 하는 모드

ESC
명령어 모드로 이동

Cursor 이동
사실 VI 에서는 방향키로 커서 이동이 안됨
K = UP
J = DOWN
H = LEFT
L = RIGHT

W = 한 단어씩 이동
B = 한 단어씩 뒤로 이동

END, $ = 현재 줄의 뒤
HOME, ^ = 현재 줄의 앞

Page Up, Ctrl + B = 한 페이지 전
Page Down, Ctrl + F = 한 페이지 다음

Ctrl + D = 반페이지씩 아래로
Ctrl + U = 반페이지씩 위로

4yy = 복사
p = 붙여넣기

. = 마지막 명령의 반복

G = 맨 끝의 라인 (파일의 맨 끝)

:1 = 첫번째 줄

명령
:w = 저장
:c = 취소
:q = 나가기
:wq = 저장 후 나가기
:cq = 저장 안하고 나가기
:q! = 닥치고 나오는거

제거
4x = 4개 제거
dd = 한줄 제거
dw = Word 단위로 제거
d$ = 혀냇 위치부터 현재 줄의 끝까지 지움
3dd = 3줄 제거

변경
~ = 소문자 -〉 대문자, 대문자 -〉 소문자
r = replace -〉 한글자만 바꿀 때
R = replace mode
cw= 한 단어를 바꾸는데 쓰는 명령

Undo
u = 방금 실행한 내용을 취소하고 뒤로 돌아감 (Ctrl + Z 같은 녀석)
U = 바로 이전꺼로 돌아감 (변경/변경취소 반복)

Redo
r = Redo (Ctrl + Y)

라인 번호 보이기
ctrl + G
set nu
set nonu

문자열 검색
'/' 〈검색어〉 (위에서 아래로)
'? ' 〈검색어〉 (아래에서 위로)
'n' 그 다음 검색 (순방향)
'N' 그 다음 검색 (역방향)

문자열 바꾸기
:범위/찾는거/바꿀내용/범위
- :s/old/new
현재 줄에서 old -〉 new (최초의 한개)
- :s/old/new/g
현재 줄에서 old -〉 new (모두)
- :10,20s/old/new/g

- :-3,+4s/old/new/g
- :%s/old/new/g
- :%s/old/new/gc

줄 붙이기
'J' - join

asdasdasd
wqdqwd

-〉

asdasdasdwqdqwd

vim
vi의 발전형

Shell

사용자가입력하는 명령을 읽고 해석하는 프로그램
- stdin, stdout 을 사용.

표준 입/출력

- 표준 입력 : stdin : 0
- 표준 출력 : stdout : 1
- 표준 에러 : stderr : 2

Redirect

- 표준 입력 : stdin : 0
- 표준 출력 : stdout : 1
- 표준 에러 : stderr : 2

Tee

자신에게 들어오는 입력 데이터를 표준 출력으로 출력하고 지정된 파일로도 보낸다.
$ sort < unsorted | tee sorted
$ sort < unsorted | tee -a sorted #append

$ tee new_file
-> 표준입력을 받고, 표준출력을하고 파일에 계속 쓴다.

환경변수

Shell이 프로그램들 사이에 어떤 값을 전달해 주는 역할을 하는 변수
기본적으로 환경 변수는 대문자를 사용
환경 변수의 값을 설정하기 위해서는 export 명령을 사용

$ export JTJ=123213213

$ printenv 또는 env, echo $PATH를 이용하여 값을 확인

$ export PATH="${PATH}:/sbin"

$ export PATH="$PATH:/sbin"

Shell Script

짧고 간단한 프로그램을 작성할 때 이용.
여러가지 쉘 명령어들을 조합하여 제작

which <파일>

명령어로 인해 실행되는 파일이 어디에 위치해있는지
("which pwd" -> /bin/pwd)