제9장 Unix/Linux의 이해

9.1 UNIX/Linux Operating System 역사

컴퓨터를 사용하기 위해서는 필수적으로 **운영체제(Operating System)**가 필요하다. 이 운영체제 중에서 가장 흔히 접할 수 있는 것은 Microsoft Windows이다. 이 외에 애플(Apple, Inc.) 사의 제품에는 MacOS X가 탑재되어 있다. 이와 같은 다양한 운영체제 중에서, 사용자가 직접 사용하는 컴퓨터가 아닌, 뒤에서 다양한 서비스를 제공해주는 **서버(Server)**를 운영하는데 가장 많이 쓰이는 운영체제는 UNIX 계열의 운영체제이다. UNIX는 1970년대에 **멀티테스킹(Multitasking)**과 **멀티유저(Multiuser)**를 지원하는 운영체제로 개발이 되었다. 그 당시에 멀티테스킹 및 멀티유저에 대한 개념은 한 대의 거대한 컴퓨터(메인 프레임 등)을 놓고, 이를 여러 사람이 터미널을 통해서 접속하여 사용하는 구조에서 시작하였다. 이를 바탕으로 여러 컴퓨터 회사에서 UNIX 운영체제를 지속적으로 개선하여 BSD(University of California, Berkeley), AIX(IBM), Xenix(Microsoft), Solaris(Sun Microsystems)등이 상업적으로 사용되고 있다 (그림 9.1).

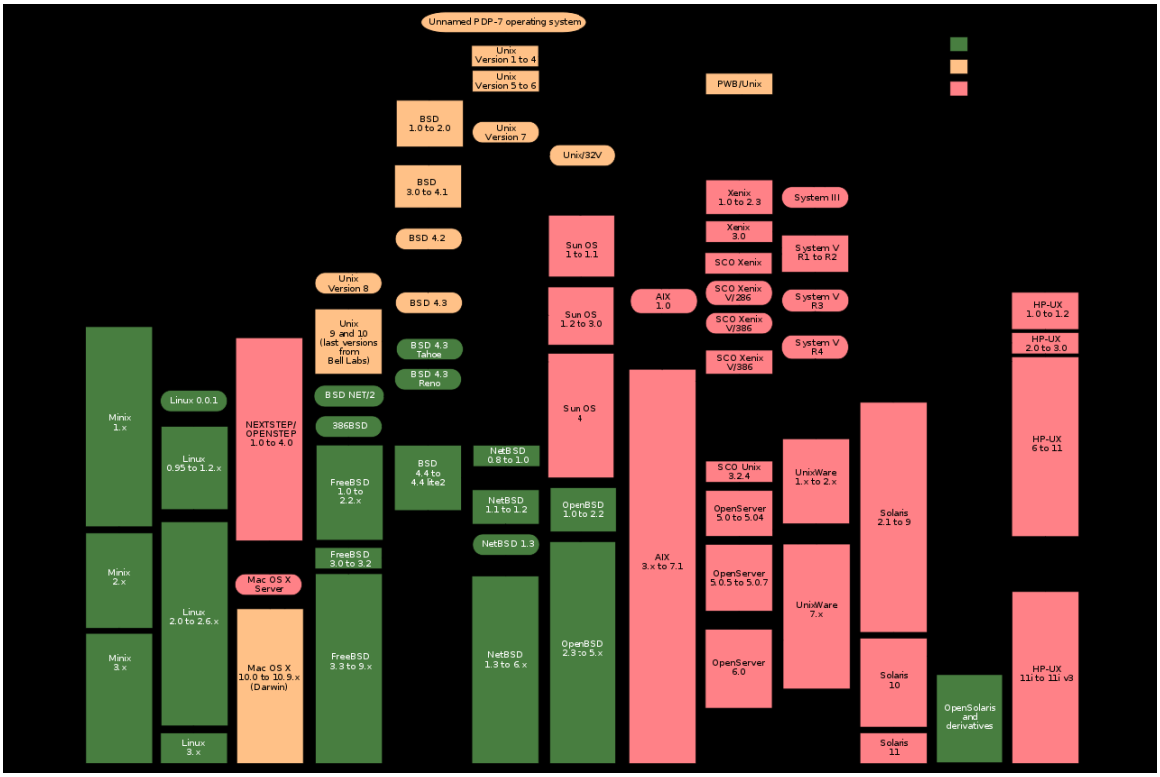


그림 9.1. 다양한 종류의 Unix 운영체제 계보도

Box 9.1. 메인프레임(Mainframe)과 터미널(Terminal).

메인프레임 (mainframe) 컴퓨터는 1950년대 후반부터 1970년대에 주로 제작이 되었다. IBM을 포함해서 Burroghs, UNIVAC, NCR, Control Data, Honeywell, General Electric, RCA 회사에서 주로 제작하였다. 위의 사진은 IBM 704 메인프레임으로 1964년에 제작이 된 사진이다. 사진 상에 있는 여성분의 키와 동일한 캐비닛 여러 개가 있을 정도로 거대한 크기를 자랑하였다. 지금의 컴퓨터와는 달리 자기테이프를 사용해서 자료를 저장하는 방식을 사용하였다.

당시에는 컴퓨터가 매우 고가의 장비였기 때문에, 컴퓨터의 자원을 최대한 활용하기 위해서 메인프레임 컴퓨터에 접속해서 사용할 수 있는 **터미널(terminal)**이 있었다. 위의 사진처럼 터미널은 모니터와 키보드가 주요 부품이며, 저장장치 및 연산은 메인프레임에서 수행될 수 있도록 설계된 컴퓨터이다. 앞으로 나올 **터미널 에뮬레이터 (terminal emulator)** 프로그램인 **Putty**도 지금 PC에서 과거 터미널을 사용하는 것과 동일한 환경을 만들어주는 역할을 담당한다.

1991년 9월에 리눅스 토발즈(Linus Torvalds)가 이 UNIX를 기반으로 하여 새로운 운영체제인 **리눅스(Linux)** 버전 0.1을 공개하고 커널(Kernal)을 포함한 모든 소스코드를 공개하였다. 초기에는 운영체제의 안정성 부분에서 문제가 있었지만, 계속적인 업그레이드를 통해서 매우 안정적인 운영체제로 자리 잡고, 대부분의 서버용 컴퓨터에서 사용이 되고 있다. 2017년 2월 22일 현재 커널은 4.10 까지 개발되었다(그림 9.2). 최근에는 동시에 빠르고 많은 연산을 하는 슈퍼컴퓨터들도 리눅스를 사용하고, 현재 TOP500 슈퍼컴퓨터 (https://www.top500.org/) 중 99% 이상이 리눅스를 탑재하고 있다.

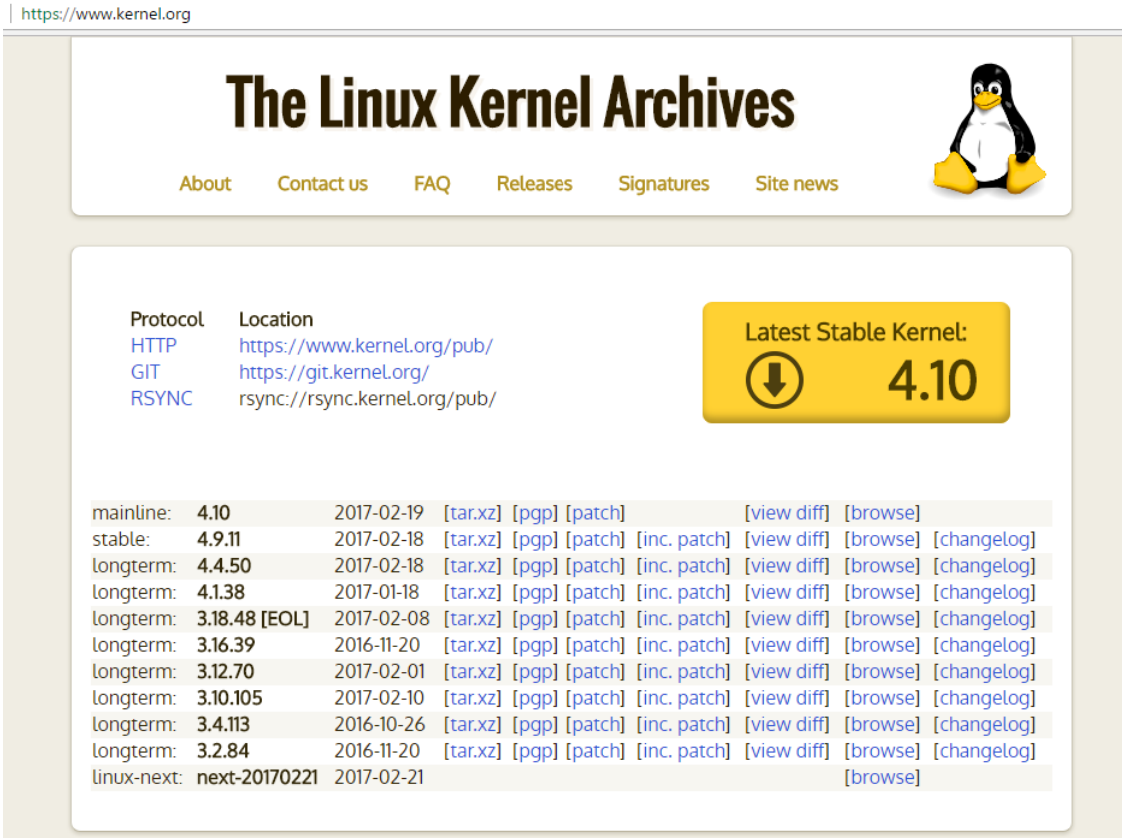
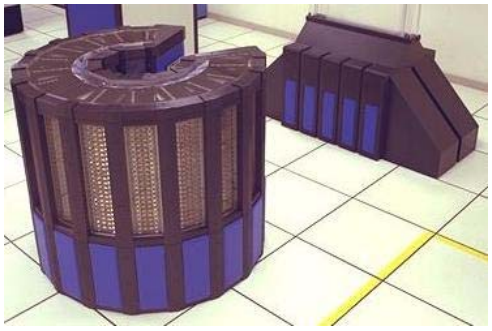


그림 9.2 최신 버전의 리눅스 커널을 제공하는 http://www.kernel.org 메인 페이지

Box 9.2. TOP500 Supercomputers.



슈퍼컴퓨터(Supercomputer)는 단어 그대로 고성능을 가진 컴퓨터를 의미한다. 과거에는 슈퍼컴퓨터 제작을 위해서 해당 컴퓨터만의 독특한 컴퓨터 구조를 사용해서 고성능을 낼 수 있도록 설계를 하였다. 대표적인 예가 크레이사(Cray Inc.)의 슈퍼컴퓨터들이다. 1972년에 크레이사가 설립되어 다양한 슈퍼컴퓨터를 만들어왔다. 크레이사의 슈퍼컴퓨터는 사진(Cray-2)처럼 일반 박스형태가 아닌 원통형 구조를 가지고 있는데, 이는 성능을 극대화하기 위해서 부품간 거리를 최소한으로 설계를 한 결과이다. 실제로 해당 컴퓨터를 제작하기 위해서 몸이 작은 여공이 직접 저 원통 안으로 가서 일일이 조립하는 방법을 사용하였다.

이 같은 단일 슈퍼컴퓨터 개념에서 1980년대 이후로는 상대적으로 성능은 떨어지는 **프로세서(processor; CPU)**를 병렬로 연결해서 보다 강력한 슈퍼컴퓨터를 만들려는 시도가 이어졌다. 이에, 현재 세계에서 가장 빠른 슈퍼컴퓨터의 대부분은 병렬 프로세서를 이용해서 구현이 되어 있다. https://www.top500.org/ 에는 2016년 11월 기준 가장 빠른 슈퍼컴퓨터의 목록을 제공하는데, 가장 빠른 슈퍼컴퓨터는 중국의 Sunway TaihuLight 으로 코어수가 무려 10,649,600개를 가지고 있고, 초당 93.0146 PFlop (9경 3014조 6천억 번의 부동소수점 연산을 수행)을 자랑한다.

TOP 10 Sites for November 2016

For more information about the sites and systems in the list, click on the links or view the complete list.

1-100 | 101-200 | 201-300 | 301-400 | 401-500

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
2	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
3	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
4	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
5	DOE/SC/LBNL/NERSC United States	**Cori** - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect Cray Inc.	622,336	14,014.7	27,880.7	3,939

이 같은 배경 때문에 Unix 및 Linux의 경우는 기본적으로 터미널(terminal) 환경을 기반으로 명령어를 주고 결과를 얻는 형태로 운영이 되고 있다. 이 방식은 PC에서 주로 사용되는 운영체제인 윈도우즈(Windows)의 **GUI (Granphical User Interface)**와 같은 편리함 대신 텍스트 기반의 환경을 기본적으로 지원하게 된다. 이는 눈으로 보고 클릭을 통해 명령을 실행하는 것이 아니라, 키보드를 통해서 직접 명령어를 입력하고 그 결과를 텍스트로 보는 방식으로 윈도우즈의 GUI에 익숙한 사용자에게는 숙달 될 때 까지 시간이 걸리기도 한다(그림 9.3).

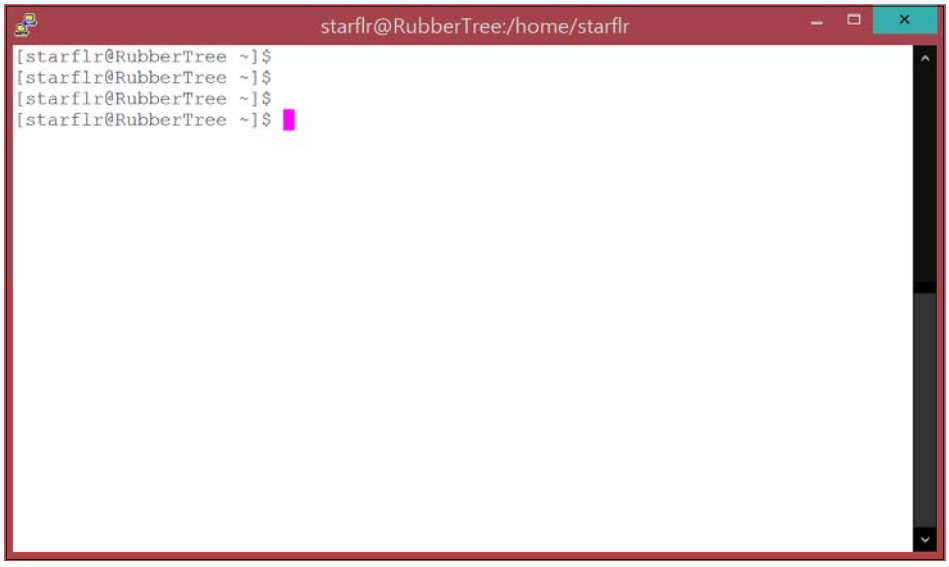


그림 9.3. Linux의 터미널 화면 모드

하지만 1970, 1980년대의 컴퓨터 환경을 고려하였을 때, 이 방법은 매우 효율적이었고, 특히 여러 사용자가 원격에서 큰 컴퓨터에 접속하여 동시에 사용할 수 있는 환경을 제공하는 이점이 있다. 1990년대 이후 컴퓨터의 성능이 계속적으로 향상됨에 따라서 Unix 기반에서도 윈도우와 같은 그래픽 환경을 제공하는 기능이 개발되었고 (X Windows System ; X11), 최근의 Linux 배포판의 경우는 예외 없이 X11을 탑재하여 쉽게 사용할 수 있도록 사용자의 편의를 제공하고 있다(그림 9.4).

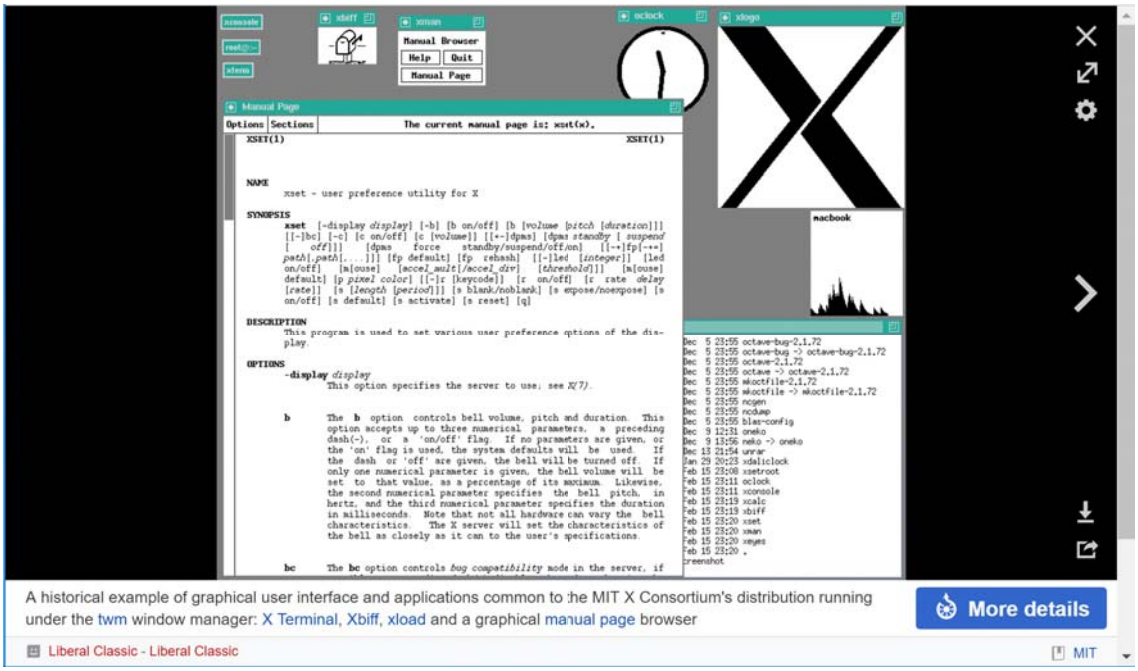


그림 9.4. X Windows System 화면 예제.

9.2 Unix/Linux과 생물정보학

운영체제의 설계는 컴퓨터 성능을 최대로 발휘시키는데 있어서 가장 중요한 부분에 속한다. Unix는 1970년대에 개발이 되었지만 멀티유저(Multiuser)와 멀티테스킹(Multitasking)을 기본 기능을 설계를 하였기 때문에, 그 당시 대형 컴퓨터(Main frame)의 자원을 여러 사람이 효율적으로 나누어서 사용할 수 있게끔 하였다(그림 9.5). 이후 Unix는 여러 가지 버전이 개발되었지만, 그 근간은 바뀌지 않았기에 DOS (Disk Operating System)를 기반으로 개발이 된 윈도우즈보다 훨씬 효율적인 운영체제로 자리 잡게 되었다. Linux또한 이 Unix를 기반으로 개발이 되어서 다양한 분석을 수행하는데 있어서 훨씬 효율적일 뿐만 아니라 무료로 배포가 되었기 때문에 초기 생물정보학 사용자들이 주로 사용하게 되었다. 이후, 대부분의 분석 프로그램들이 Linux에서 개발이 되어서 배포되고 있다.

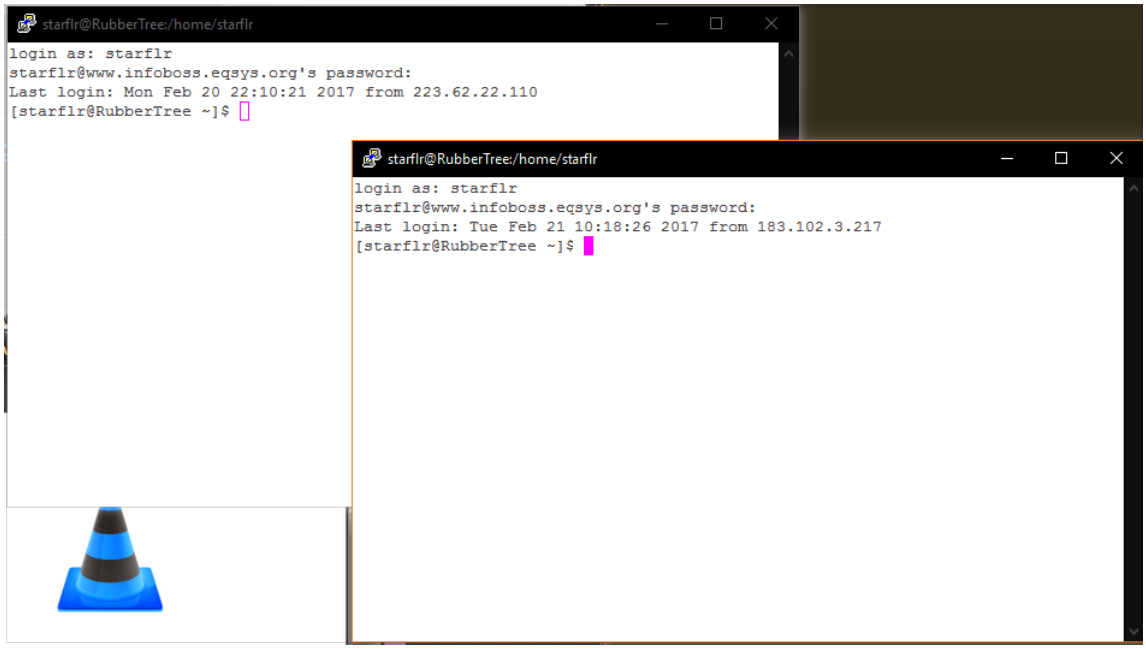


그림 9.5 Linux는 기본적으로 Multitasking을 지원한다.

추가적으로 애플 컴퓨터의 경우는 MacOS 9 버전까지는 자체적인 운영체제를 개발 및 운영하여 왔으나,

MacOS X 부터는 운영체제를 완전히 새로 개발하면서 Unix 기반의 커널 중 하나인 Mach 커널을 기반으로 개발하였다. 이는 간접적으로 Unix가 얼마나 안정적인지를 보여주는 또 다른 예이다. 지금 판매되는 모든 맥 노트북 및 데스크탑은 모두 Unix 기반의 운영체제에 의해서 구동된다고 보아도 무방하다.

9.3 Unix/Linux 시작하기

Unix/Linux를 사용하기 위해서는 운영체제에 **계정(acocunt)이** 있어야 한다. 이는 운영체제를 사용하기에 앞서서 내가 누구인지 인증을 하는 과정이다. 참고로 윈도우즈의 경우도 여러 사람이 사용할 때는 계정을 여러 개 만드는 경우가 있는데, 이것과 동일한 기능으로 보면 된다. 계정 생성은 시스템 관리자에게 문의를 하여 발급을 받도록 한다.

계정이 준비되었으면 서버로 접속을 해야 한다. 보통 서버의 접속은 인터넷을 통해서 진행을 하므로, **IP 주소 (IP address)** 혹은 **도메인 이름(domain name)**이 필요하다. 이 주소와 함께, 어떠한 프로토콜 (protocol)로 접속을 할 것인지에 대한 정보가 필요하다. 이 프로토콜이란 서버와 통신하는 방법을 정의한 것으로 과거에는 telnet을 많이 사용하였고, 지금은 보안이 강화된 **SSH (Secure SHell)**을 주로 사용한다.

> **Box 9.3. IP주소 (IP Address).**
>
> IP는 Internet Protocol의 약자로, 인터넷에 연결된 컴퓨터의 고유 주소를 의미한다. 현재 널리 사용되고 있는 IP 주소 구조는 IPv4로 총 4개의 숫자로 이루어져 있다. 이론적으로는 최대 $256^4 = 2^{32} = 4{,}294{,}967{,}296$개의 주소를 표현할 수 있다. 하지만 몇몇 IP 주소는 특수한 용도로 사용이 되는데, 192.168.x.x 는 **내부망 (private network)**에서 사용되는 수치가 그 예이다.
>
> 최근 들어서, 다양한 종류의 모바일 기기, 사물인터넷 기기들이 출현하면서 IP 주소가 충분하지 않게 되어 IPv6 기반으로 보다 많은 기기의 고유번호를 가질 수 있는 **규약(protocol)**이 제정되었다. 우리나라의 경우는 아직 IPv6는 거의 보급 되지 않고 있고 IPv4를 사용하고 있다.

윈도우 환경에서 Unix/Linux 서버에 접속하기 위해서는 **터미널 에뮬레이터(Terminal Emulator)**가 필요하다. 본 장에서는 **Putty** 프로그램을 기반으로 설명하도록 한다. 이 프로그램은 무료 프로그램이며 http://www.putty.org/ 에서 다운로드가 가능하다. 현재 버전은 0.67 이다. 프로그램 설치 후 실행을 시키면 아래 그림과 같은 화면이 나타난다(그림 9.6). 여기에 IP주소를 입력하고 SSH를 선택하여 서버에 접속을 한다.

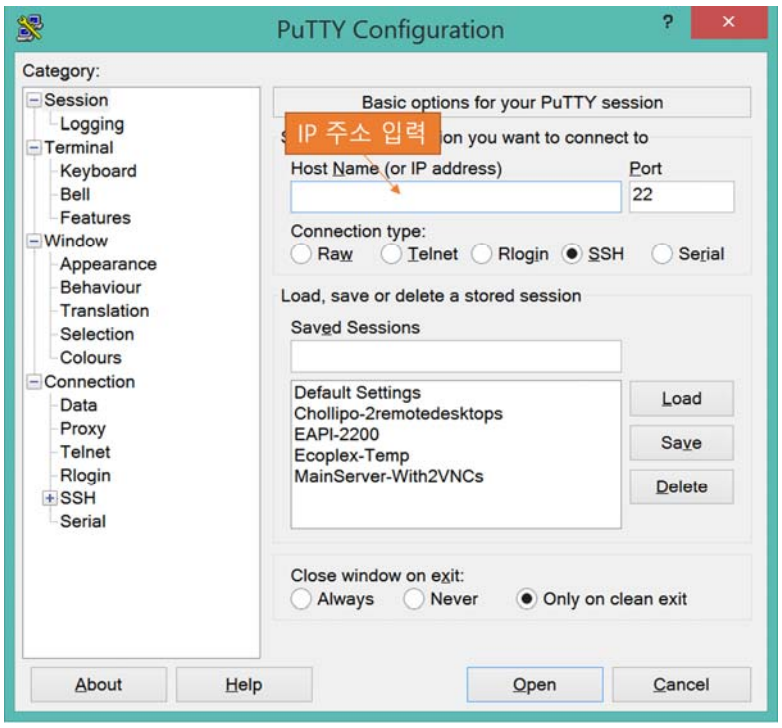


그림 9.6. 무료 터미널 프로그램인 Putty의 첫 화면.

정상적으로 접속이 되면 ID와 암호를 묻는 과정을 거치게 된다(그림 9.7). 정상적인 아이디와 암호를 입력하게 되면 가장 마지막에 **프롬프트(Prompt)**가 출력된다. 이는 서버가 현재 명령을 대기하고 있음을 알려주는 공통된 신호이다. 앞의 부분의 문자는 Linux 배포판에 따라서 다양하게 나타날 수 있지만 가장 마지막의 '$'는 공통적으로 보여주는 부분이다. 만약 이 $대신에 #이 있는 경우는 관리자(superuser) 권한으로 접속이 된 것임을 알려준다. 드디어 Unix/Linux 운영체제가 설치된 서버에 접속을 하였다.

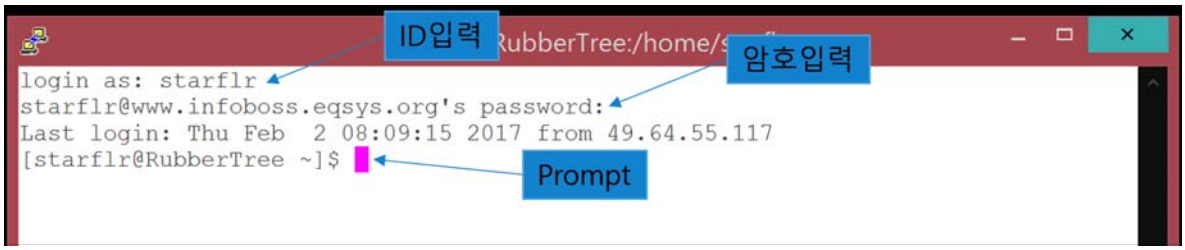


그림 9.7. 서버 접속에 성공한 화면.

9.4 Unix/Linux 명령어

프롬프트가 정상적으로 나오게 되면 서버에 명령을 주어서 원하는 동작을 시킬 수 있는 상태이다. 이 프롬프트에 명령어를 입력하면 서버는 해당 명령어에 대한 결과를 출력해준다. Unix/Linux는 윈도우즈와 다르게 이름의 대소문자를 구분함에 유의한다.

9.4.1 ls

Unix/Linux의 공통적인 명령어 중에 현재 위치의 파일을 보여주는 명령어이다. 명령어 다음에 한 칸 띄고 옵션을 줄 수 있는데, 아래 화면의 예제에 있는 -al 옵션은 숨김 파일을 포함한 모든 파일 (-a)을 출력하고, 각 파일에 대한 구체적인 정보를 리스트 형태로 (-l) 보여 달라고 하는 것을 의미한다(그림 9.8).

명령어 1
ls -al

그림 9.8. ls -al 명령의 결과 화면.

위의 화면에서 보면 첫 줄과 두 번째 줄에 .과 .. 이 파란색으로 나타나는데, 이는 특수 폴더(directory)를 의미한다. 첫 번째 '.'은 현재 폴더를 의미하는 것이고, 두 번째 '..'은 상위 폴더를 의미하는 것이다. 두 번째 명령어인 cd를 이용해서 현재 폴더를 변경하고자 할 때 이 특수 폴더를 사용할 수 있다. 그 다음에 .으로 시작하는 파일이 4개가 나오는데, 파일 이름이 '.'으로 시작을 하면 Linux에서는 숨김 파일에 해당된다. 이에, -a 옵션을 빼면, '.'으로 시작하는 파일이 출력되지 않는다(그림 9.9).

그림 9.9. ls -l의 결과 화면.

9.4.2 cd 및 pwd

cd는 현재 위치를 변경하는 명령어로 change directory의 약자이다. 명령어 다음에 원하는 폴더 (directory) 이름을 입력하면 해당 폴더로 이동하게 된다(그림 9.10). 상위 폴더로 이동시에는 'cd ..'을 입력하면 된다 (9.4.1의 특수 폴더 참조).

명령어 2

cd [directory name]

명령어 3

cd ..

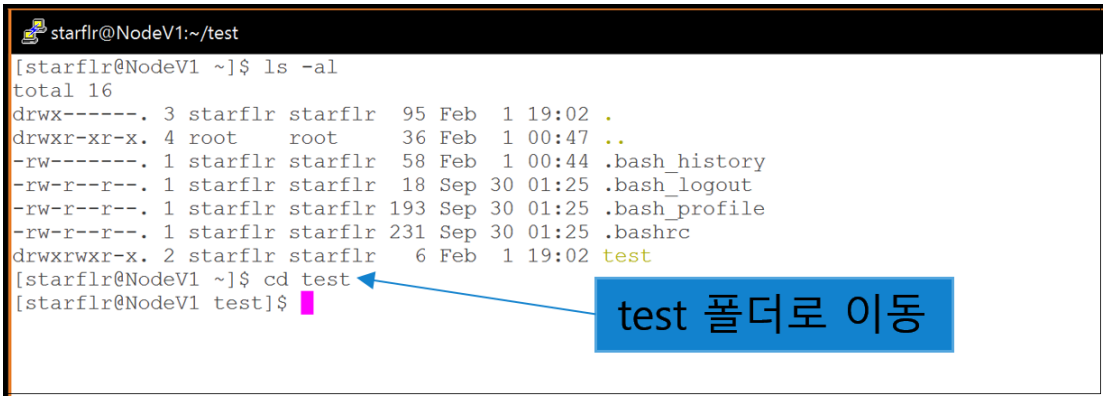


그림 9.10. cd 명령어를 통한 폴더 이동 결과.

pwd는 현재 경로를 출력하는 명령어로 자신의 경로를 아래와 같이 표시해준다(그림 9.11). /home/starflr/test 중에서 앞 부분인 /home/starflr 는 본 계정의 홈 폴더이다.

명령어 4

pwd

그림 9.11. 현재 경로 출력(pwd).

9.4.3 clear

현재 보이는 화면을 지울 때 쓰는 명령어이다. 터미널에 나와 있는 글씨가 모두 소거된다(그림 9.12).

명령어 5

clear

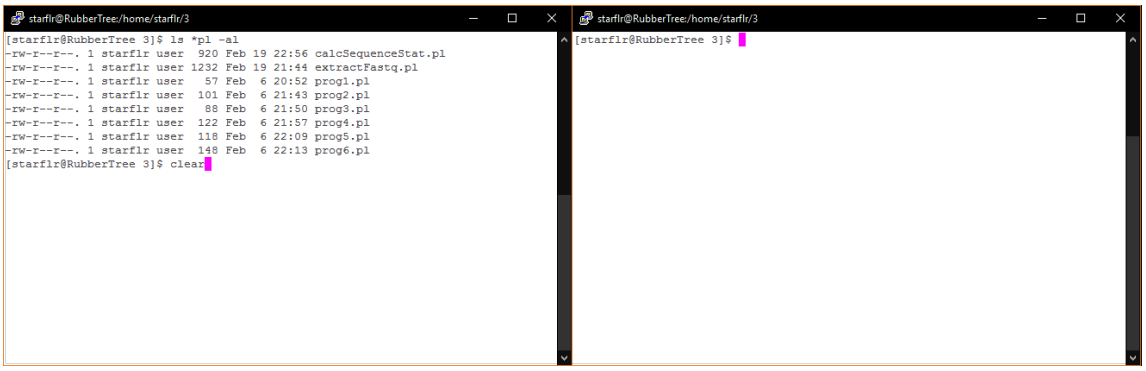


그림 9.12 clear 명령 전 후 화면 변화

9.4.4 cat

특정 파일의 내용을 화면으로 출력하는 명령어이다. cat 다음에 파일 이름을 입력하면 해당파일 내용이 출력된다(그림 9.13). 아래 예제는 test.txt 파일의 내용을 출력하기 위해서 cat 명령을 사용한 경우이다. ls -al 명령을 통해서 현재 폴더에 test.txt 파일이 있음을 확인한 후에 cat test.txt를 입력하여 안의 내용을 출력하였다.

명령어 6

cat [filename]

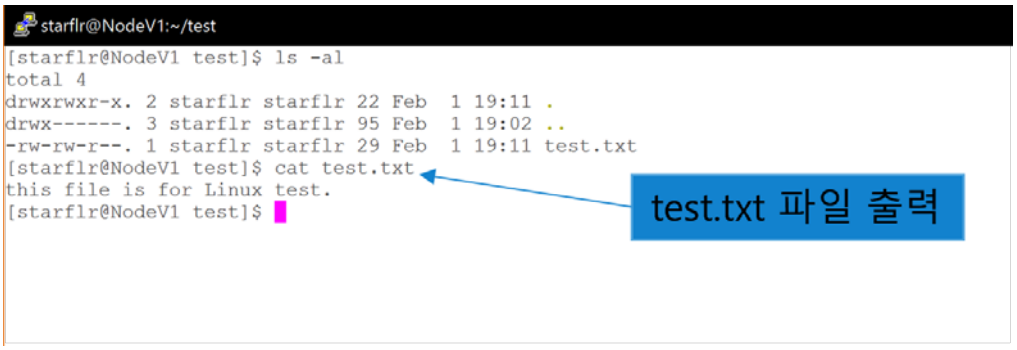


그림 9.13. cat 명령을 통해 test.txt 파일의 내용을 화면에 출력하였다.

연속적으로 여러 개의 파일 내용을 출력하고자 할때는 원하는 파일 이름을 계속 써주면 된다. 예를 들면, cat test1.txt test2.txt 하면 두 개의 파일을 순차적으로 화면에 출력해준다.

명령어 7

cat test1.txt test2.txt

```
starflr@RubberTree:/home/starflr/3
[starflr@RubberTree 3]$ cat test1.txt
This is test1.txt file.
[starflr@RubberTree 3]$ cat test2.txt
This is test2.txt file.
[starflr@RubberTree 3]$ cat test1.txt test2.txt
This is test1.txt file.
This is test2.txt file.
[starflr@RubberTree 3]$
```

그림 9.14 cat test1.txt test2.txt 실행 예제

Box 9.4. 두 개의 파일 합치기 (Redirect)

Linux에서는 command line을 이용해서 매우 다양한 일들을 수행할 수 있다. cat 프로그램은 단순하게 파일 내용을 출력하는 역할을 하는데, 이를 이용해서 2개의 파일을 1개의 파일로 합치는 것도 가능하다.

cat test1.txt test2.txt 명령어 결과를 보면 2개의 파일 내용이 순차적으로 나오는 것을 확인할 수 있는데(그림 9.14), 이 결과를 파일로 다시 쓰면 합쳐진 파일을 생성할 수 있게 된다.

이를 위해서 아래와 같이 명령어에 한가지를 더 추가해보자.

명령어 8

cat test1.txt test2.txt > test3.txt

>는 redirect를 의미하는데, 이는 > 왼쪽에서 화면에 출력되는 결과를 오른쪽 파일에 쓰라는 의미이다. 좀 더 엄밀하게 말하면 왼쪽 명령어의 STDOUT (화면 출력)의 결과를 파일 test3.txt에 쓰라는 의미가 된다.

test3.txt 내용을 확인하면 아래와 같다.

```
starflr@RubberTree:/home/starflr/3
[starflr@RubberTree 3]$ cat test1.txt test2.txt > test3.txt
[starflr@RubberTree 3]$ cat test3.txt
This is test1.txt file.
This is test2.txt file.
[starflr@RubberTree 3]$
```

2개의 파일 내용이 하나로 합쳐져서 test3.txt에 잘 보관되었다.

9.4.5 grep

Unix/Linux에서는 text 파일을 다룰 수 있는 많은 프로그램이 준비되어 있다. 9.4.4의 cat도 그 중에 하나이고, grep은 지정한 파일(들)에서 특정한 문자열이 존재하는지를 찾는 프로그램이다. 사용 예제는 아래와 같다 (그림 9.15).

명령어 9

grep Linux test1.txt

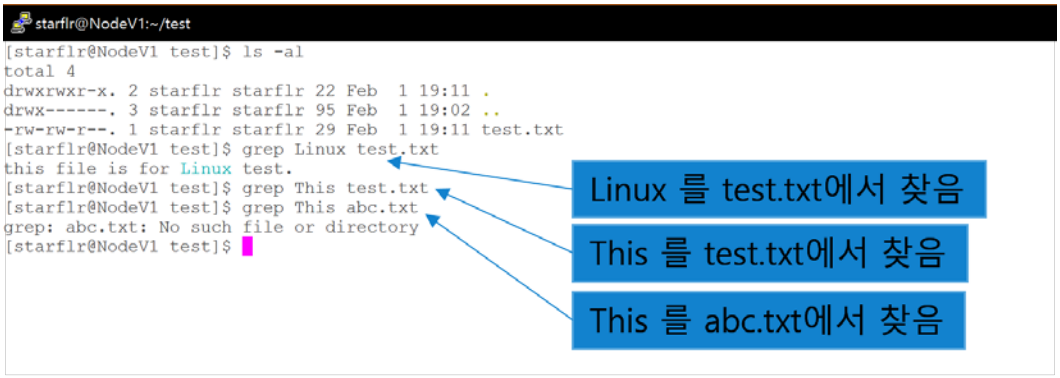


그림 9.15. grep 사용방법 예제.

'grep [문자열] [파일이름]' 형태로 입력하면 [파일이름]의 파일에서 [문자열]이 있는 경우 해당 문자열이 포함된 줄을 출력해준다. grep Linux test.txt 의 경우 해당 파일 안에 Linux가 있으므로 해당 줄이 출력되고 Linux가 빨간색으로 표시되었다. 두 번째 명령어에서는 This를 찾는 명령어이나, 결과가 없으므로 아무것도 출력되지 않았다. 마지막으로 없는 파일이름을 지정하는 경우는 파일이 없다는 에러 메시지가 출력된다.

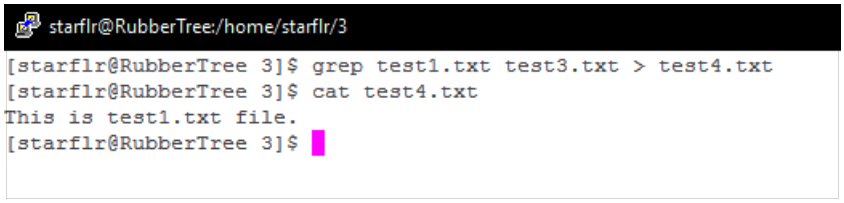
Box 6.5. 파일내용 일부분만 추출해서 새로운 파일 만들기

Box 6.4에서 파일을 합치는 예제를 cat과 redirect를 사용하여 수행하였다. 이번에는 grep을 이용해서 특정 파일의 부분만을 추출하여 새로운 파일을 만들어 보는 것을 진행해본다.

명령어 10

grep test1.txt test3.txt > test4.txt

위의 명령어는 test3.txt에서 test1.txt 라는 문자열이 있는 줄만을 추출(grep)하여 test4.txt 파일에 쓰라는 의미이다. 해당 명령어를 실행시켜보면 아래와 같은 결과가 나타난다.



9.4.6 pipe (|)

Unix/Linux에서는 간단한 명령어들을 조합해서 여러 가지 기능을 할 수 있는 명령을 만드는 것이 가능하다. Box 9.4와 Box 9.5에서 소개한 redirect가 그런 예중의 하나이다. 이와 비슷하게, 서로 다른 프로그램을 연결

하는 방법으로 Linux에서는 파이프(pipe) 라는 방법을 제공하고 있다(그림 9.16). 이는 프로그램에서 결과로 나오는 내용을 다음 프로그램의 입력으로 처리되도록 두 프로그램을 동시에 구동하여 처리하게 된다. 이 방법은 간단한 프로그램을 연속적으로 연결해서 원하는 결과를 쉽게 얻을 수 있도록 도와준다. 파이프를 정의할 때는 "|" 문자를 사용한다.

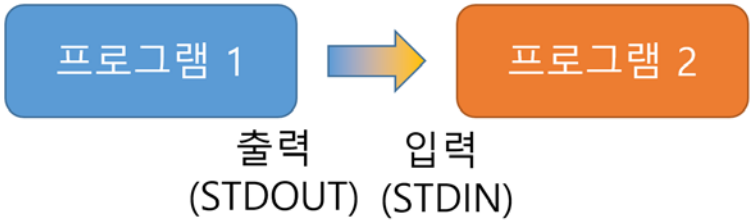


그림 9.16. Unix 파이프 구조.

파이프는 두 개 이상의 프로그램을 연속적으로도 연결할 수 있다. 예를 들어서, 특정 파일(test2.txt)에서 Linux와 This가 동시에 들어있는 줄을 찾고자 하는 경우에는 9.4.5의 grep 명령어로 한번에 결과를 얻을 수 없게 된다. 하지만, 처음에 Linux를 찾는 명령(grep Linux test2.txt)를 사용한 후에 다시 그 결과를 바탕으로 This를 찾게 된다면, 원하는 결과를 얻을 수 있게 된다. 또 다른 예제로, ls -al 의 결과에서 test2.txt 라는 문자열을 골라내고 싶을 때 ls -al 명령어의 출력을 grep 명령의 입력으로 사용하게 되면 원하는 결과를 얻어낼 수 있다.

명령어 11

grep Linux test2.txt | grep This

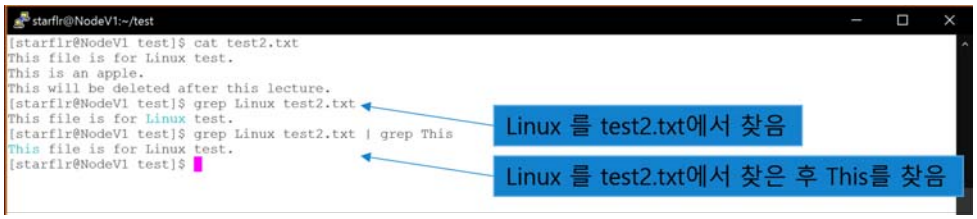


그림 9.17. pipe를 이용하여 두 개의 문자열을 동시에 검색하는 예제.

|(파이프 기호)를 첫 번째 명령어 다음에 쓴 두 번째 명령어 grep에서는 파일 이름을 명시하지 않는다. 왜냐하면, 입력이 파일이 아니라 첫 프로그램의 결과가 될 것이기 때문이다. 대부분의 Unix 프로그램은 입력 소스를 명시하지 않으면 STDIN, 즉 사용자로부터의 입력을 대기하게 되는데, 파이프는 이 입력(STDIN)에 이전 프로그램의 결과를 넣어주는 역할을 하여, 두 번째 프로그램은 첫 번째 프로그램의 결과를 기반으로 해당 기능을 수행하게 된다.

이와 같이, 두 개 이상의 프로그램을 연결해서 사용하게 되면 복잡한 명령을 새로운 프로그램을 만들지 않고 바로 해결할 수 있는 장점이 있고, 실제 생물정보학 분석에서도 파이프는 매우 빈번하게 사용되고 있다.

Box 9.6. STDIN, STDOUT, STDERR

Box 9.1에서 메인프레임과 터미널에 대해서 소개를 하였다. 터미널에서는 **사용자의 입력** (STDIN; Standard Input)을 받아서 (키보드로부터) 그 내용을 처리하고 결과를 **화면으로 출력**

(STDOUT; Standard output) 하는 기능을 수행하게 된다. 여기에 프로그램의 에러나 경고의 경우는 STDERR로 출력이 되게 된다. 기본적으로 STDOUT과 STDERR는 모두 화면에 표시되기 때문에 눈으로 보기에는 큰 차이가 없으나, redirect를 이용해서 두 가지 출력은 구분이 가능하다.

명령어 12

```
ls -al > output.txt 2> error.txt
```

위의 명령어를 보면 ls -al의 결과를 output.txt에 저장하도록 하고, STDERR로 출력되는 메시지는 error.txt 로 저장되도록 하는 명령어이다. redirect에서 >는 STDOUT에서 나온 결과를 저장하는 것을 의미하고 2> 는 STDERR로 나오는 문자를 저장하는 것을 의미한다.

9.4.7 vi editor

Linux에서 주로 사용하는 텍스트 에디터(text editor)로는 vi와 emac가 가장 대표적이다. Linux 사용자는 보통 이 두 개의 에디터중 하나를 주로 사용하는데, 본 교재에서는 vi 에 대해서 간단하게 소개를 하려 한다.

Box 9.7. Editor War

Unix 계열 운영체제에서 가장 많이 쓰이는 text editor는 vi와 emacs이다. 거의 대부분의 사용자가 둘 중 한 개의 에디터를 사용하다보니 두 그룹 사이에서는 미묘한 긴장감이 항상 흐르게 된다. 이를 가리켜서 'Editor war'라는 표현을 사용한다.

(https://en.wikipedia.org/wiki/Editor_war)

서버에 접속된 상태에서 vi 에디터로 새로운 파일(파일이름 : test3.txt) 을 만들고 저장해보도록 한다. 먼저 vi test3.txt 명령어를 입력한다(그림 9.18).

그림 9.18. vi 에디터 구동 명령어 입력 : test3.txt 파일을 새로 만든다.

명령어 입력 후에 vi 에디터 화면이 나타난다. 새로운 파일을 만드는 것이므로, 하단에 [New File] 이라는 메시지가 뜨고 빈 화면이 나타난다. 여기에서는 더 이상 프롬프트가 나타나지 않는데, 이는 vi 에디터가 실행중임을 의미한다(그림 9.19).

그림 9.19. vi 에디터 실행 화면.

vi 에디터는 일반적인 워드 프로세서와 매우 다른 개념이 있다. 지금 보는 화면에서는 실제 원하는 텍스트를 바로 입력할 수 있는 것이 아닌 명령어를 입력하는 모드이다. 이 모드에서는 커서의 이동, 파일의 저장, 복사 및 붙여넣기 등의 기능을 키보드만으로 수행한다(터미널 프로그램은 윈도우의 화면과 완전하게 동일하지 않으므로 붙여넣기, 마우스 휠을 이용한 스크롤 등을 시도하고자 할 때 혼동이 있을 수 있다).

텍스트 입력을 위해서 소문자 i를 입력하면 화면 하단에 —INSERT—가 나타나는 것을 볼 수 있다. 이 글씨가 보이면 자유롭게 글씨를 입력할 수 있다(그림 9.20).

그림 9.20. vi 에디터의 입력 모드.

테스트로 “This is an example of vi editor under Linux.” 문장을 입력해보았다. 다음 줄로 이동하려면 Enter 키를 누르면 다음 줄로 이동하게 된다. 이 상태에서 커서 이동은 화살표 키로 가능하며, 글씨 삭제는 Backspace 키 혹은 Del 키를 사용하면 된다(그림 9.21).

그림 9.21. vi 에디터에서 한 문장을 입력한 화면.

입력이 완료되면 ESC 키를 누른다. 이는 입력모드에서 탈출한다는 의미이며, 실제로 하단의 —INSERT— 가 사라진 것을 볼 수 있다. 이 상태에서 :w를 입력하고 Enter키를 누르면 파일이 저장된다. 파일이 성공적으로 저장되면 하단에 파일 이름과 문자의 총 개수(45C)가 함께 출력이 된다(그림 9.22).

그림 9.22. vi 에디터에서 파일을 저장한 경우.

파일 저장이 끝났으므로 vi 에디터를 종료해보도록 한다. 종료하는 방법은 입력 모드가 아닌 현 상태에서 :q 를 입력하면 종료된다. q는 quit의 약자이다(그림 9.23).

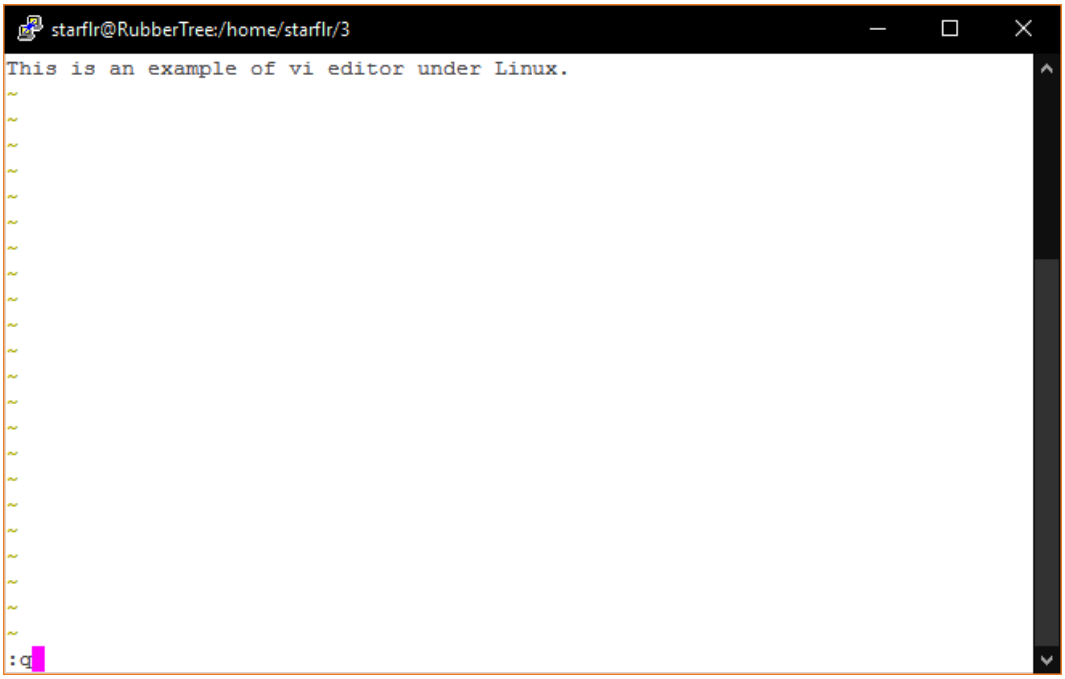


그림 9.23 vi editor 종료를 위한 명령어 입력 화면

파일을 생성하였으므로 해당 파일이 잘 만들어졌는지를 ls 명령어로 확인하고 해당 내용을 cat 프로그램으로 확인해보도록 한다(그림 9.24)

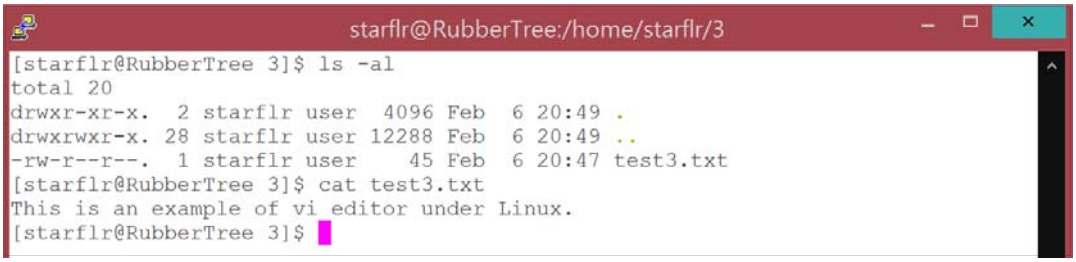


그림 9.24. vi 에디터로 생성한 파일 확인 및 내용 확인.

위와 같이 파일이 잘 생성되었고 내용이 입력한 것과 동일한 것을 확인할 수 있다.

Box 9.8. vi editor와 화살표 키

지금의 키보드는 대부분 101키를 근간으로 만들어져 있다. 88키 구조와는 달리 오른쪽에 숫자 키패트가 추가가 된 구조이며, 보통 숫자에는 화살표(왼쪽 4, 위쪽 8, 오른쪽 6, 아래쪽 2)가 그려져 있고, 키패드 왼쪽으로 친절하게 4개의 화살표 키가 제공된다.

과거에는 이렇게 친절한 화살표 키가 존재하지 않다보니, 영문자를 이용해서 상하좌우를 이동하곤 하였다. 이때 사용된 문자는 h (왼쪽), l (오른쪽), j (윗쪽), k (아랫쪽)을 의미한다. 이 키는 vi editor의 명령어 모드에서 커서의 위치를 이동시킬 때 여전히 사용이 되며, 화면 분할 후에 화면간 이동에서도 사용이 된다.

9.4.8 w

Linux 명령어들은 대부분 간결하게 정의가 되어 있는데, w의 경우도 그에 해당한다. w 명령은, 현재 접속된 사용자에 대한 정보를 보여준다. 처음에서도 언급했듯이 Linux는 multiuser를 근간으로 하기 때문에 동시에 여러 사용자가 한 서버에 접속해서 사용하는 것이 기본 기능이 된다. 이에 따라서 현재 접속한 접속자에 대한 정보도 쉽게 조회할 수 있다. 또한 한 개의 계정으로 여러 개의 접속이 가능한데, 아래 그림 9.25에서 보면 한 개의 아이디어 여러 번 접속해서 서로 다른 명령어를 실행하고 있음을 알 수 있다.

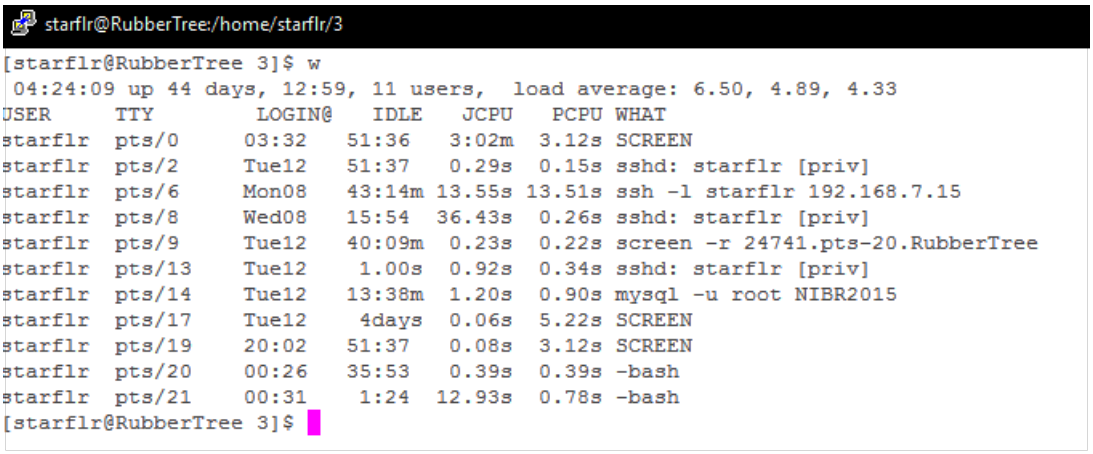


그림 9.25 현재 접속한 사용자 정보 조회

9.4.9 exit or logout

현재 사용 중인 세션(session)을 종료하고자 할 때 exit 혹은 logout 명령을 통해서 종료를 할 수 있다 (그림 9.26). 물론 Putty 프로그램을 그냥 종료해도 되지만, 명시적으로 종료를 하는 것이 더 좋다.

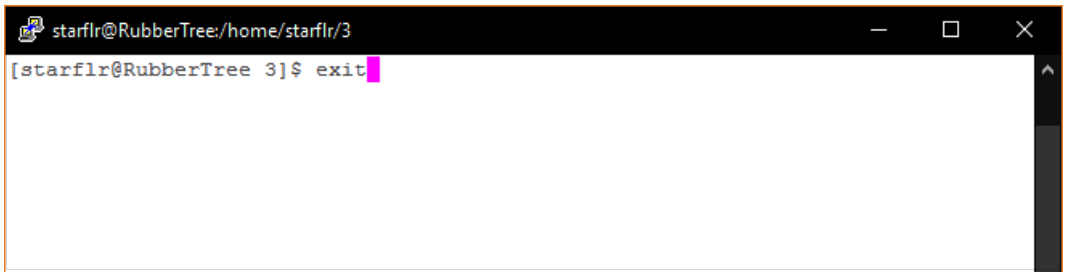


그림 9.26. quit 명령 화면