웹프로그래밍의 기초

Week2

리눅스 기본 명령어 및 vi 에디터 사용법

Why Text Interface for now?

Mainframe and Terminal

- Unix, one of the early operating systems, was designed to run as a multi-user system on mainframe computers, with users connecting to it remotely via individual ***terminals***.
- Just a keyboard and screen, with no power to run programs locally. Instead they would just send keystrokes to the server and display any data they received on the screen.
- Compared with graphics, text is very light on resources.
- The commands were very terse to reduce the number of keystrokes needed, speeding up people's use of the terminal even more.
- Speed and efficiency is one reason why this text interface is still widely used today.

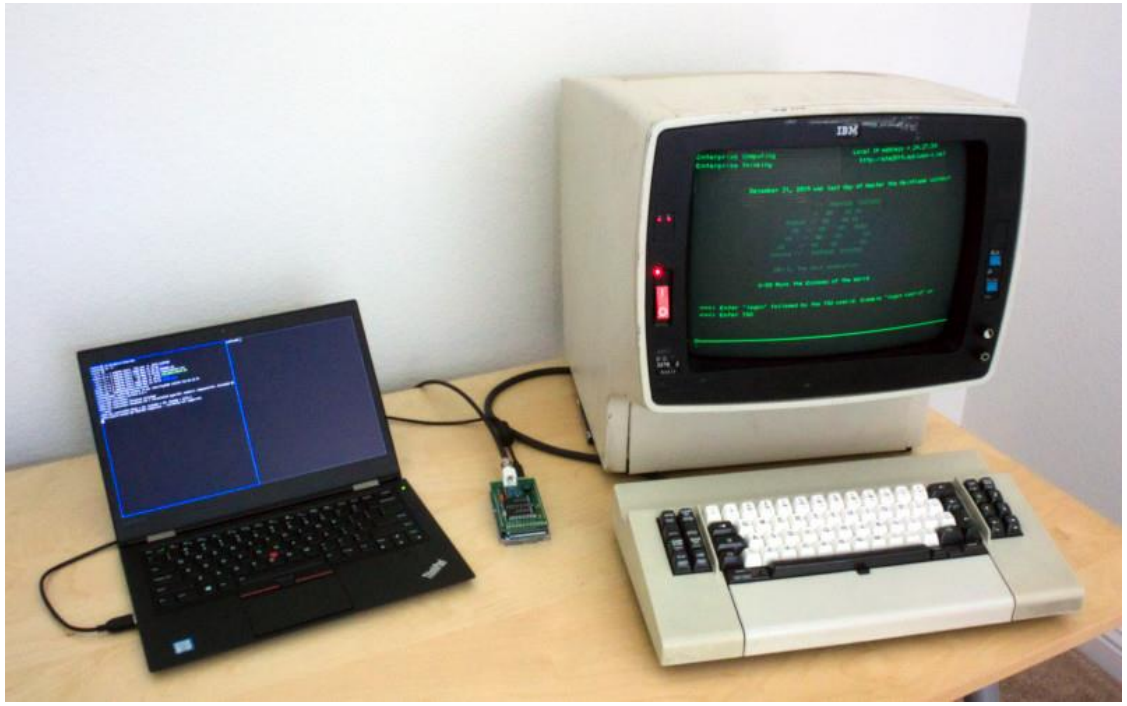
The original Unix shell program was just called sh.

Shell

- In order to coordinate the execution of each management task program, the user would connect to one single master program that could then be used to launch any of the others.
- By wrapping the user's commands ***this "shell" program***, as it was known, could provide common capabilities to any of them, such as the ability to pass data from one command straight into another, or to use special wildcard characters to work with lots of similarly named files at once.
- Users could even write simple code (called ***"shell scripts"***) which could be used to automate long series of shell commands in order to make complex tasks easier.
- The original Unix shell program was just called ***sh(Bourne shell)***, a modern Linux system you're most likely to be using a shell called ***bash(Bourne Again Shell).***

Shell Compatibility

- Linux is a sort-of-descendent of Unix.
- The core part of Linux is designed to behave similarly to a Unix system, such that most of the old shells and other text-based programs can run on. In theory you could even hook up one of those old 1970s terminals to a modern Linux box, and access the shell through that
- But these days it's far more common to use a software terminal: that same old Unix-style text interface, but running in a window alongside your graphical programs.



Bringing those into the user's hands

Presenting the IBM® of Personal Computers.

IBM is proud to announce a product *you* may have a personal interest in. It's a tool that could soon be on your desk, in your home or in your child's schoolroom. It can make a surprising difference in the way you work, learn or otherwise approach the complexities (and some of the simple pleasures) of living.

It's the computer we're making for you.

In the past 30 years, the computer has become faster, smaller, less complicated and less expensive. And IBM has contributed heavily to that evolution.

Today, we've applied what we know to a new product we believe in: the IBM Personal Computer.

IBM PERSONAL COMPUTER SPECIFICATIONS
*ADVANCED FEATURES FOR PERSONAL COMPUTERS

User Memory
16K - 256K bytes*
Permanent Memory
(ROM) 40K bytes*
Microprocessor
High speed, 8088*
Auxiliary Memory
2 optional internal diskette drives, 5¼", 160K bytes per diskette
Keyboard
85 keys, 6 ft. cord attaches to system unit*
10 function keys*
10-key numeric pad
Tactile feedback*

Display Screen
High-resolution (720h x 350v)*
80 characters x 25 lines
Upper and lower case
Green phosphor screen*
Diagnostics
Power-on self testing*
Parity checking
Languages
BASIC, Pascal
Printer
Bidirectional*
80 characters/second
12 character styles, up to 132 characters/line*
9 x 9 character matrix*

Color/Graphics
Text mode:
16 colors*
256 characters and symbols in ROM*
Graphics mode:
4-color resolution: 320h x 200v*
Black & white resolution: 640h x 200v*
Simultaneous graphics & text capability*
Communications
RS-232-C interface
Asynchronous (start/stop) protocol
Up to 9600 bits per second

It's a computer that has reached a truly personal scale in size and in price: starting at less than $1,600† for a system that, with the addition of one simple device, hooks up to your home TV and uses your audio cassette recorder.

For flexibility, performance and ease of use, no other personal computer offers as many advanced features to please novice and expert alike (see the box).

Features like high resolution color graphics. Ten, user-defined function keys. The kind of expandability that lets you add a printer for word processing, or user memory up to 256KB. Or BASIC and Pascal languages that let you write your own programs. And a growing list of superior programs like VisiCalc,™ selected by IBM to match the quality and thoughtfulness of the system's total design.

This new system will be sold through channels which meet our professional criteria: the nationwide chain of 150 ComputerLand® stores, and Sears Business Systems Centers. Of course, our own IBM Product Centers will sell and service the system. And the IBM Data Processing Division will serve those customers who want to purchase in quantity.

Experience the IBM Personal Computer. You'll be surprised how quickly you feel comfortable with it. And impressed with what it can do for you.

IBM®

The IBM Personal Computer and me.

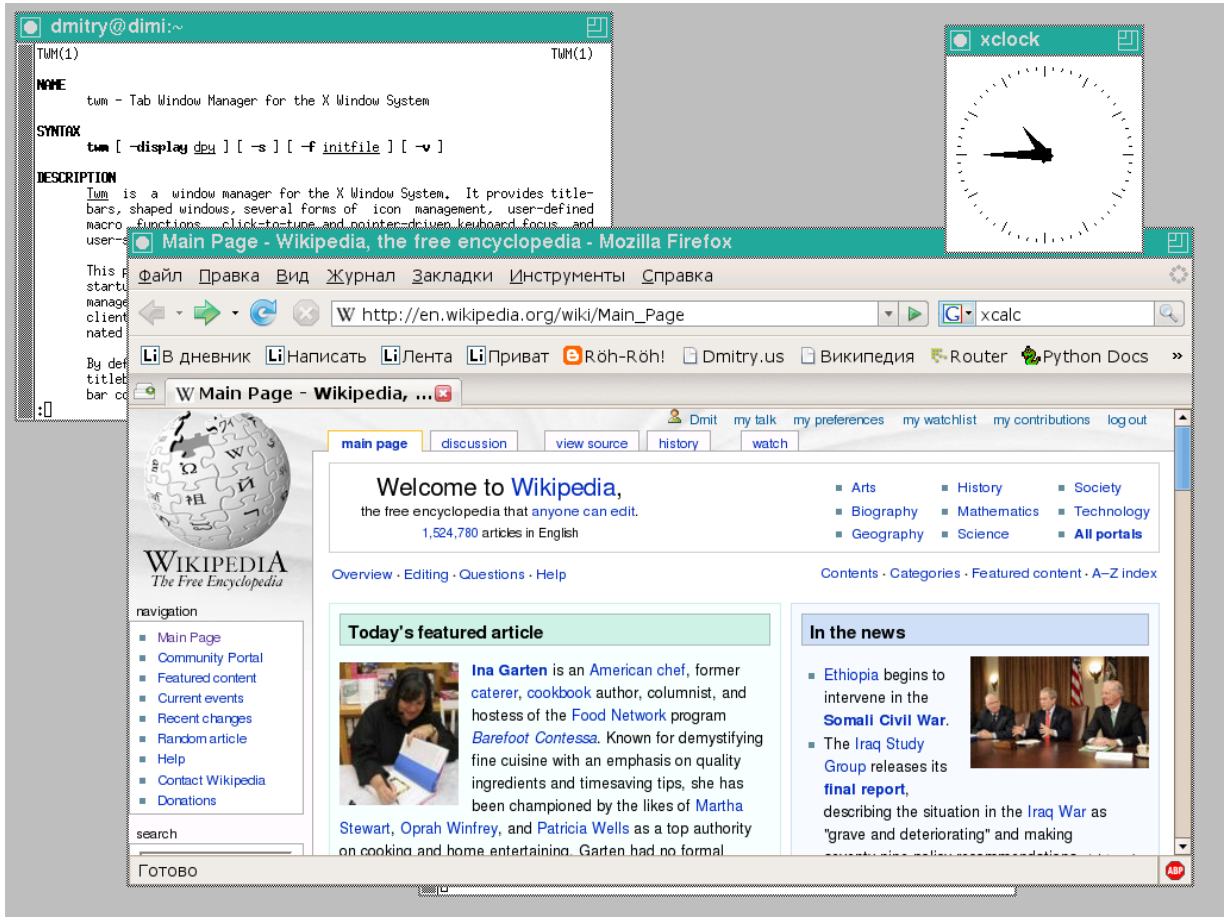
For the IBM Personal Computer dealer nearest you, call (800) 447-4700. In Illinois, (800) 322-4400.

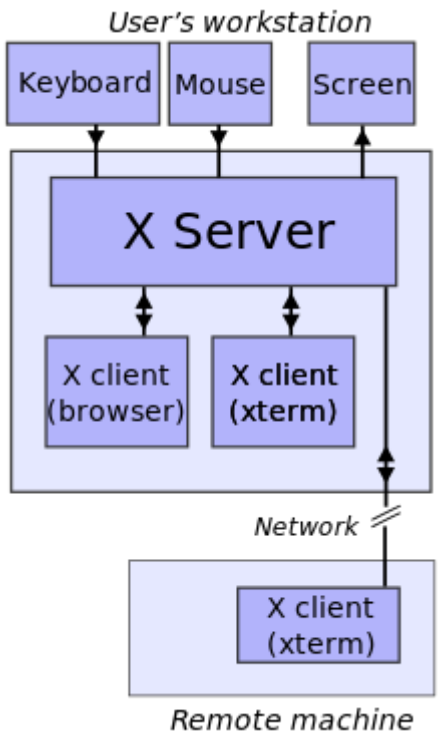
CIRCLE 3

†This price applies to IBM Product Centers. Prices may vary at other stores.
VisiCalc is a trademark of Personal Software, Inc.

1981

X Window System, or X11





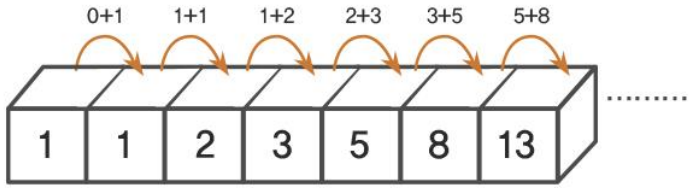
X originated as part of Project Athena at Massachusetts Institute of Technology (MIT) in 1984
The X protocol has been at version 11 (hence "X11") since September 1987.

And Innovation to GUI from CLI

1984

All the programming interface is based on Text input & output

```
≡ File Edit Search Run Compile Debug Project Options Window Help
[■] ELI\CPP_ST~1\OLD_C_~1\BORLAND\MYPROGS\FIBONACI.C 3=[↕]
#include <stdio.h>
#include <conio.h>

int i, j, inpt;
ar[20];

main()
{
  clrscr();

  printf("Enter number (1 to 20) ? ");
  scanf("%d",& inpt);

  ar[0] = ar[1] = 1;
  printf("\n 1 1");

  for (i = 2; i <= inpt; i++)
  {
    ar[i] = ar[i-1] + ar[i-2];
    printf(" %d",ar[i]);
  }
1:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu
```



Basic Linux Commands

Creating folders and files

Moving and manipulating files

Basic Linux commands

- cd 는 디렉토리를 탐색하는 데 사용됩니다. 경로를 통해 모든 위치로 이동할 수 있습니다.
- ls 는 폴더 내용 나열에 사용됩니다. 다양한 종류의 파일과 폴더 속성을 모두 볼 수 있습니다.
  - ls -l - 소유자, 사용 권한, 크기 및 수정된 날짜를 포함하는 더 긴 목록을 제공합니다.
  - ls -a - 숨겨진 파일 및 폴더와 일반 목록을 표시합니다.
  - ls -al - 두 옵션을 결합하여 숨겨진 파일과 폴더를 모두 표시하고 긴 형식으로 표시합니다.
- cp 는 파일 복사에 사용됩니다.
  - cp file /path/to/folder - 지정된 파일을 지정된 경로에 복사합니다.
- mv 는 파일 이동에 사용됩니다.
  - mv file /path/to/folder - 지정된 파일을 지정된 경로에 이동합니다.
- rm 은 파일 제거에 사용됩니다.
  - rm file - 시스템에서 해당 파일을 제거합니다.
  - rm -r folder - 시스템에서 해당 폴더를 제거합니다.
- mkdir 은 디렉토리 만들기에 사용됩니다.
  - mkdir folder_name - 지정한 이름으로 폴더를 만듭니다.
- > 및 >> 리디렉터는 터미널 대신 파일로 출력 전송에 사용됩니다.
  - > 는 새 명령의 출력으로 대체하여 기존 파일 내용을 overwrite하는 데 사용됩니다.
  - >> 는 기존 파일에 정보를 append하는 데 사용됩니다. 이 명령은 로깅 작업에 유용합니다.`

Pathnames

- (홈 디렉토리) This directory is owned by each user of the system.
  - Changing directory to user home directory

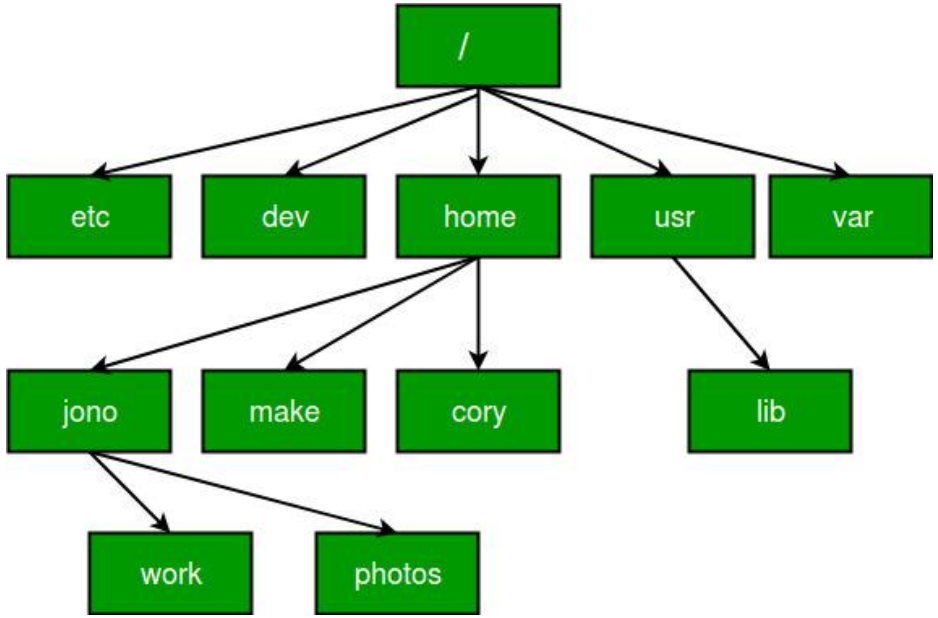
    cd
    Or cd ~
    Or CD $HOME

- (절대경로) Absolute path is defined as specifying the location of a file or directory from the root directory(/).
  - Changing directory with absolute path concept:

    $pwd
    /home/kt
    $cd /home/kt/abc
    $pwd
    /home/kt/abc

- (상대경로) Relative path is defined as the path related to the present working directly(pwd).
  - .(a single dot) - this represents the current directory.
  - ..(two dots) - this represents the parent directory.
  - Changing directory with relative path concept :

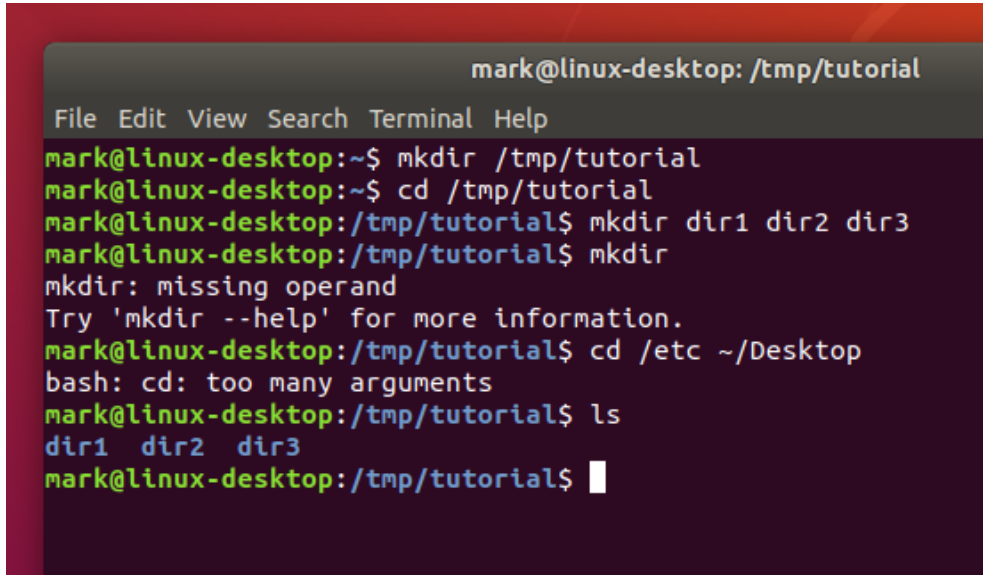
    $pwd
    /home/kt
    $cd abc
    $pwd
    /home/kt/abc

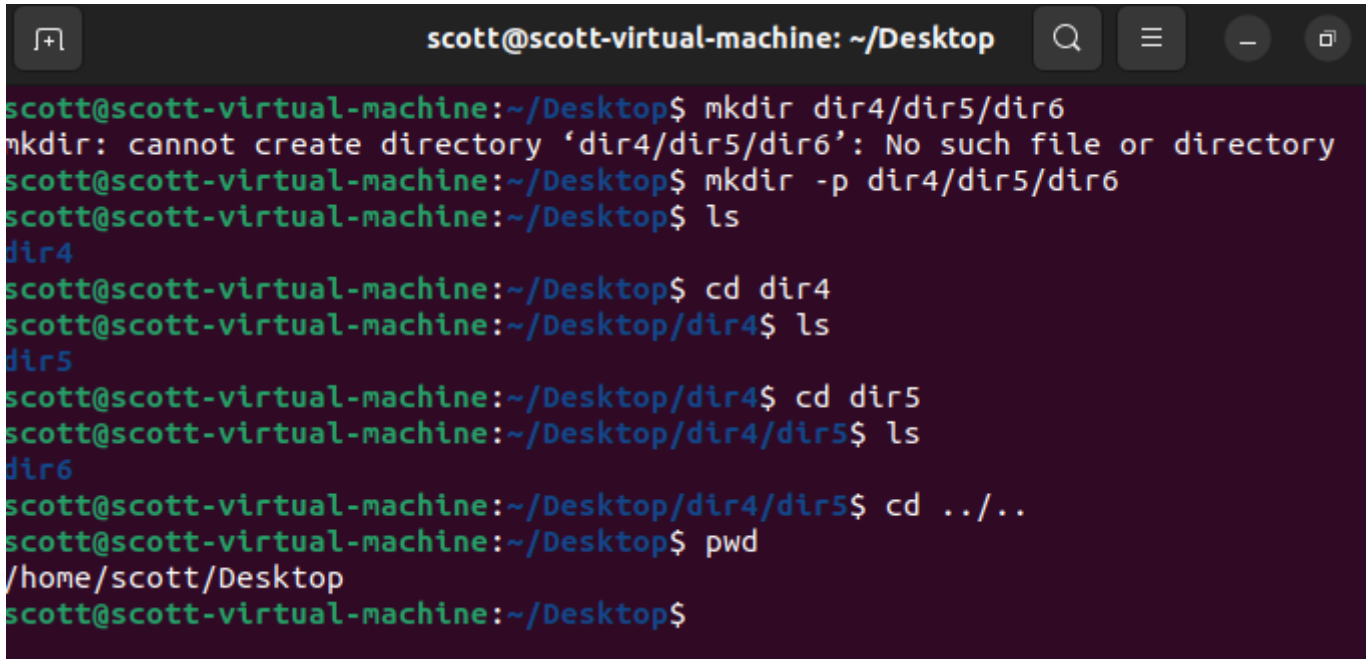


Creating folders

```
mkdir /tmp/tutorial
cd /tmp/tutorial
mkdir dir1 dir2 dir3
mkdir
cd /etc ~/Desktop
ls
```

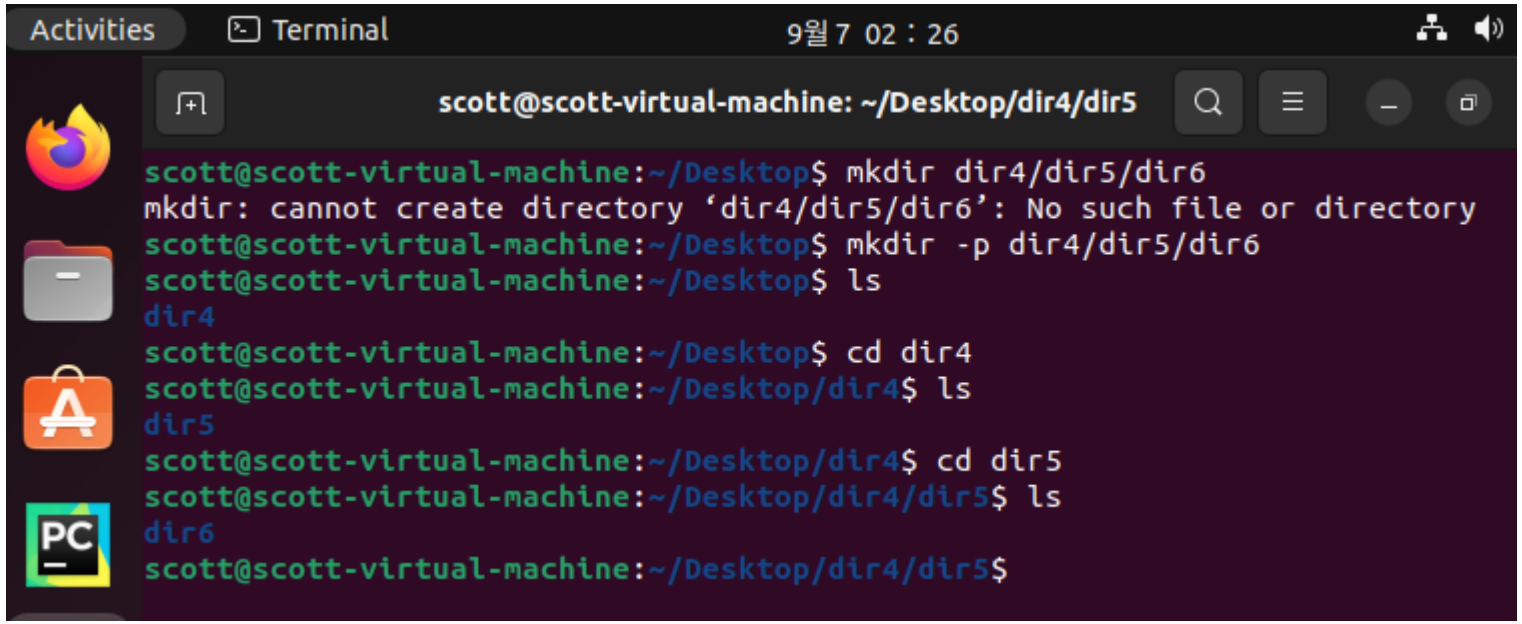
```
mkdir -p dir4/dir5/dir6
ls
cd dir4
ls
cd dir5
ls
cd ../..
```





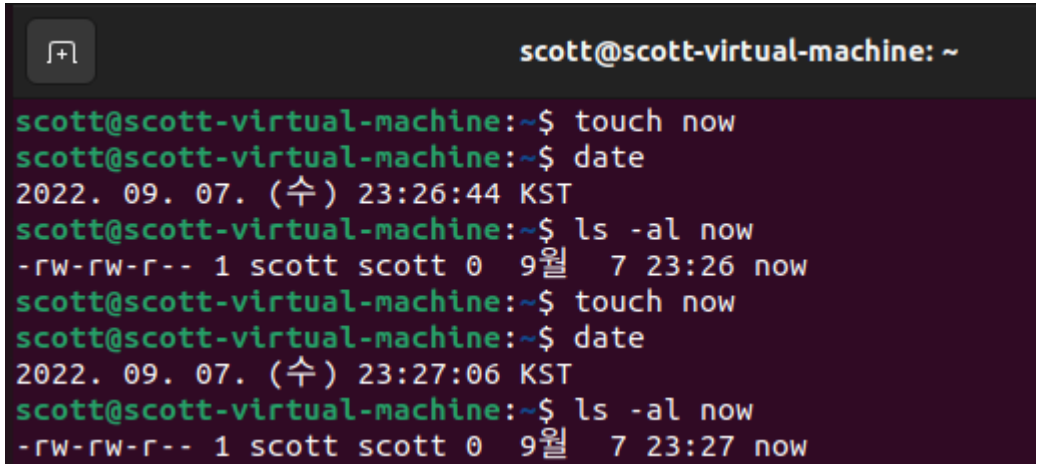
Creating folders

```
mkdir -p dir4/dir5/dir6
ls
cd dir4
ls
cd dir5
ls
cd ../..
```



Creating files

- The touch command comes as part of the GNU Core-utilities and creates a new file in Linux using the terminal. The touch command's primary function is to modify a timestamp. Commonly, the utility is used for file creation, although this is not its primary function.



Creating files

Oprator > and >>

```
scott@scott-virtual-machine:~$ ls > listoffile.txt
scott@scott-virtual-machine:~$ cat listoffile.txt
Desktop
Documents
Downloads
listoffile.txt
Music
Pictures
Public
snap
Templates
Videos
scott@scott-virtual-machine:~$ ls >> listoffile.txt
scott@scott-virtual-machine:~$ cat listoffile.txt
Desktop
Documents
Downloads
listoffile.txt
Music
Pictures
Public
snap
Templates
Videos
Desktop
Documents
Downloads
listoffile.txt
Music
Pictures
Public
snap
Templates
Videos
scott@scott-virtual-machine:~$
```

Creating files

- Using cat commands, multiple files can be merged in to the on file.

```
scott@scott-virtual-machine: ~
scott@scott-virtual-machine:~$ echo "life of a" > a.txt
scott@scott-virtual-machine:~$ echo "life of b" > b.txt
scott@scott-virtual-machine:~$ cat a.txt
life of a
scott@scott-virtual-machine:~$ cat b.txt
life of b
scott@scott-virtual-machine:~$ cat a.txt b.txt > ab.txt
scott@scott-virtual-machine:~$ cat ab.txt
life of a
life of b
scott@scott-virtual-machine:~$
```

Moving and Manipulating both directories and files

```
scott@scott-virtual-machine:~$ ls
ab.txt  b.txt    Documents  Music     Public  Templates
a.txt   Desktop  Downloads  Pictures  snap    Videos
scott@scott-virtual-machine:~$ mv ab.txt a_b.txt
scott@scott-virtual-machine:~$ ls
a_b.txt  b.txt    Documents  Music     Public  Templates
a.txt    Desktop  Downloads  Pictures  snap    Videos
scott@scott-virtual-machine:~$ rm a_b.txt
scott@scott-virtual-machine:~$ ls
a.txt  Desktop    Downloads  Pictures  snap       Videos
b.txt  Documents  Music      Public    Templates
scott@scott-virtual-machine:~$ mkdir somthingin
scott@scott-virtual-machine:~$ mkdir nothingin
scott@scott-virtual-machine:~$ touch somthingin/somethin
scott@scott-virtual-machine:~$ rmdir nothingin/
scott@scott-virtual-machine:~$ rmdir somthingin/
rmdir: failed to remove 'somthingin/': Directory not empty
scott@scott-virtual-machine:~$ ls
a.txt  Desktop    Downloads  Pictures  snap        Templates
b.txt  Documents  Music      Public    somthingin  Videos
scott@scott-virtual-machine:~$ rm -fr somthingin/
scott@scott-virtual-machine:~$ ls
a.txt  Desktop    Downloads  Pictures  snap       Videos
b.txt  Documents  Music      Public    Templates
scott@scott-virtual-machine:~$
```

Vi Editor

Opening and saving files

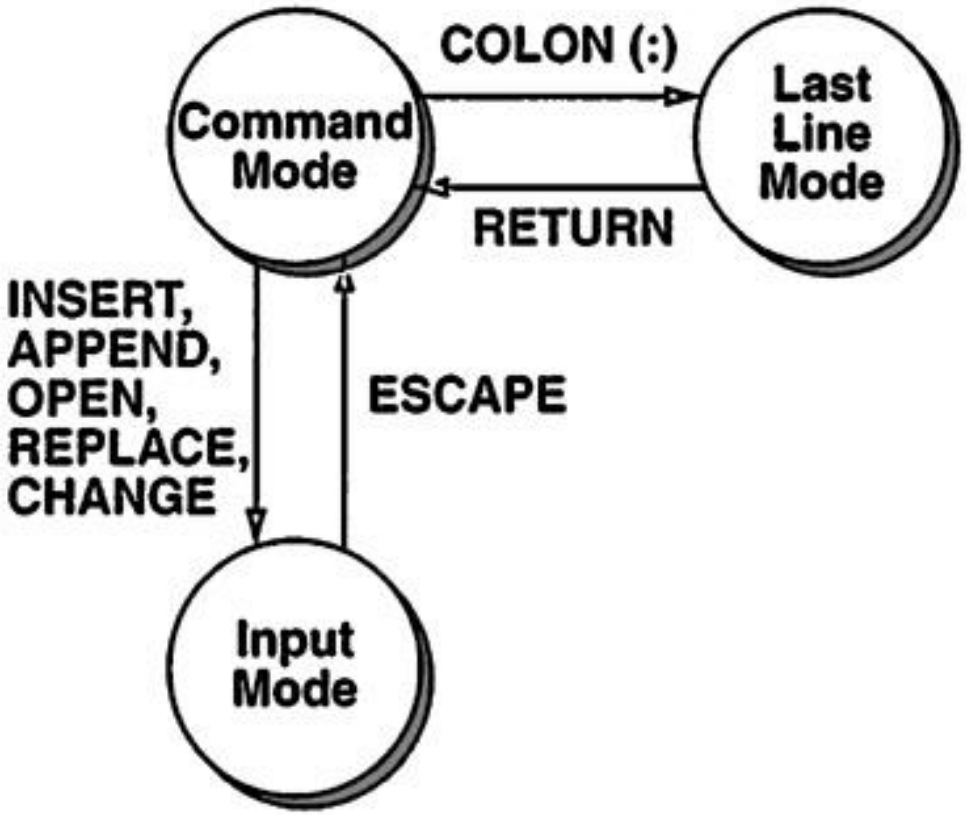
Editing files

Vi guides

- https://www.redhat.com/sysadmin/introduction-vi-editor
- https://www.marquette.edu/mathematical-and-statistical-sciences/basic-vi-editor-commands.php#:~:text=Command%20mode%20is%20the%20mode,%2C%20%2C%20key.
- https://www.cs.colostate.edu/helpdocs/vi.html

Vi modes

- vi command mode
  - When you first start editing a file with the vi editor you will be in vi command mode. In this mode you can issue many vi commands, including commands like insert, append, and delete, and other search and navigation commands that let you move around your file.
- vi insert mode
  - Once you issue a vi insert, append, or open command, you will be in vi insert mode. You want to switch back to vi command mode, you easily move back to command mode by pressing the [Esc] key.
- vi last line mode
  - The last vi mode is known as vi last line mode. You can only get to last line mode from command mode, and you get into last line mode by pressing the colon key, like this:
  - ":set showmode" will show "-- INSERT --" line if you are in insert mode.



Use Google for your own specific tasks

- https://kb.iu.edu/d/acoj

University Information Technology Services

About Services Initia

Knowledge Base Menus About the team

Search the Knowledge Base... Log in

ARCHIVED: In vi, how can I perform a global search and replace?

This content has been archived, and is no longer maintained by Indiana University. Information here may no longer be accurate, and links may no longer be available or reliable.

To perform a global search and replace in vi, use the search and replace command in command mode:

```
:%s/search_string/replacement_string/g
```

The `%` is a shortcut that tells vi to search all lines of the file for `search_string` and change it to `replacement_string`. The global (`g`) flag at the end of the command tells vi to continue searching for other occurrences of `search_string`. To confirm each replacement, add the confirm (`c`) flag after the global flag.

At Indiana University, for personal or departmental Linux or Unix systems support, see Get help for Linux or Unix at IU.

Related documents

Use the vi text editor

This is document acoj *in the Knowledge Base.*
Last modified on 2018-01-18 09:27:02.

Other vi references

- https://www.redhat.com/sysadmin/introduction-vi-editor
- https://www.cs.colostate.edu/helpdocs/vi.html
- https://docs.oracle.com/cd/E19683-01/806-7612/6jgfmsvq7/index.html