

# Lab 1

---

*Data Structure*

# Git Usage

## 1. Set user info

```
$ git config --global user.name "2016000000"
```

```
$ git config --global user.email "2016000000@hanyang.ac.kr"
```

# Git Usage

## 2. Clone project

```
$ git clone http://hconnect.hanyang.ac.kr/2017_[Course no.]_20XXXXXXXXX.git
```

You can check the address of your git at GitLab webpage. GitLab 사이트에서 주소 확인 가능



# Git Usage

## 2. Clone project

And type your username and password

```
[YSPARKui-MacBook-Air:~ YS$ git clone http://hconnect.hanyang.ac.kr/2017_CSE2010_13058/2017_CSE2010_0000000000.git
Cloning into '2017_CSE2010_0000000000'...
Username for 'http://hconnect.hanyang.ac.kr': 2016101704
[Password for 'http://2016101704@hconnect.hanyang.ac.kr':
warning: You appear to have cloned an empty repository.]
```



## *Lab 2*

1. Sign in a new GitLab server
2. Do Lab 2-1, Lab 2-2, and Lab 2-3

# Evaluation criteria

Category	Evaluation	
2-1	33	
2-2	33	
2-3	33	
Total	100	

- *Use GCC 4.8 version. No score will be given if the gcc version is different.*

# *GCC*

- *Version Check*

```
$ gcc --version
```

- *Install*

```
$ sudo apt-get install gcc-4.8
```

- *Change Version*

```
$ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.8 80
```

# Linux Command

Command	Description
ls	List directory contents.
ls -l	List directory contents in long listing format showing permissions, ownership, size, and time and date stamp.

```
$ ls
```

```
$ ls [Directory Path]
```

```
$ ls -l
```



# Linux Command

Command	Description
mkdir	Create a new directory
cd	Change directory

```
$ mkdir [Directory name]
```

```
$ cd [Directory Path]
```

# Linux Command

Command	Description
rm	remove files and directories.

```
$ rm [file name]
```

```
YSPARKui-MacBook-Air:DS_Test YS$ ls
dir1      dir2      file1     file2
YSPARKui-MacBook-Air:DS_Test YS$ rm file1
YSPARKui-MacBook-Air:DS_Test YS$ ls
dir1      dir2      file2
```

# Linux Command

Command	Description
rm	remove files and directories.

```
$ rm -r [directory name]
```

```
YSPARKui-MacBook-Air:DS_Test YS$ ls
dir1    dir2    file2
YSPARKui-MacBook-Air:DS_Test YS$ rm dir1
rm: dir1: is a directory
YSPARKui-MacBook-Air:DS_Test YS$ rm -r dir1/
YSPARKui-MacBook-Air:DS_Test YS$ ls
dir2    file2
```

# Linux Command

Command	Description
cp	Copy files and directories.
mv	Rename or move file(s) or directories.

```
$ cp [Source name] [Destination]
```

```
$ mv [Source name] [Destination]
```

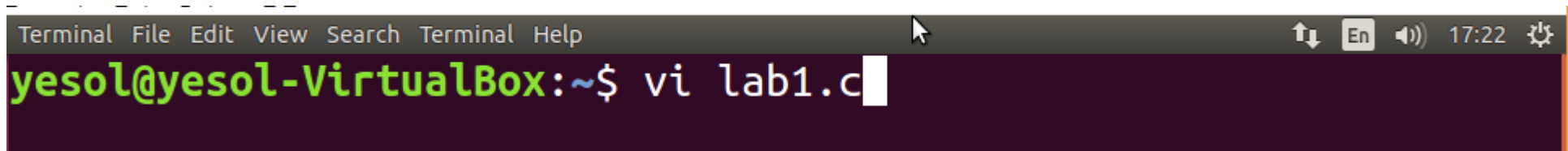


## *Lab2*

- You should finish the first problem (Lab 2–1, Lab 2–2, Lab 2–3) during the lab session and submit it to the git. You can leave for home after confirm it with the TA.
- Folder name : Lab2

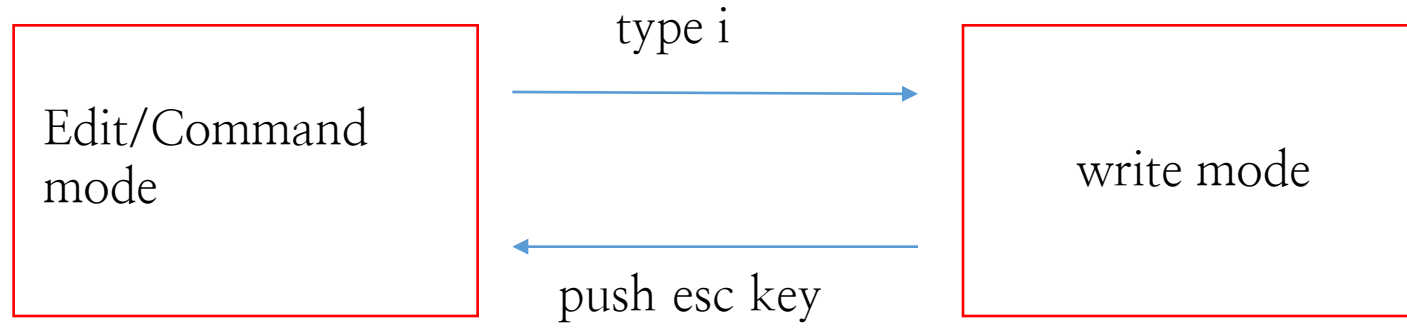
# *Vi text editor*

- To start : `vi [filename]`

A screenshot of a terminal window with a dark purple background. The window has a title bar with 'Terminal' and a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. On the right side of the title bar are icons for window management, a keyboard layout indicator showing 'En', a speaker icon, and the time '17:22'. The terminal prompt is 'yesol@yesol-VirtualBox:~\$' in green text. The command 'vi lab1.c' is entered in white text, followed by a white cursor.

- Three mode : Write / Edit / Command line
- There are several commands to enter “Write” mode, but normally we type “i”.

# *Vi text editor*



# *Vi text editor*

- Useful commands in **edit mode**

x          Delete on character(at cursor)

dd        Delete on row

yy        Copy a one row

p          Paste

uu        Cancel(undo) latest job

Ctrl+r    Redo

uG        Move cursor to n-th row

gg / G    Move cursor to first/last row



# *Vi text editor*

- Useful commands in **command line mode**

Every first symbol of the command is the accessor to command line mode (for example, '/' or ':')

/[keyword] : find [keyword] starting from cursor

→ n / N : move cursor to next / previous entry

:wq                      Save & exit

:q                        Exit without save

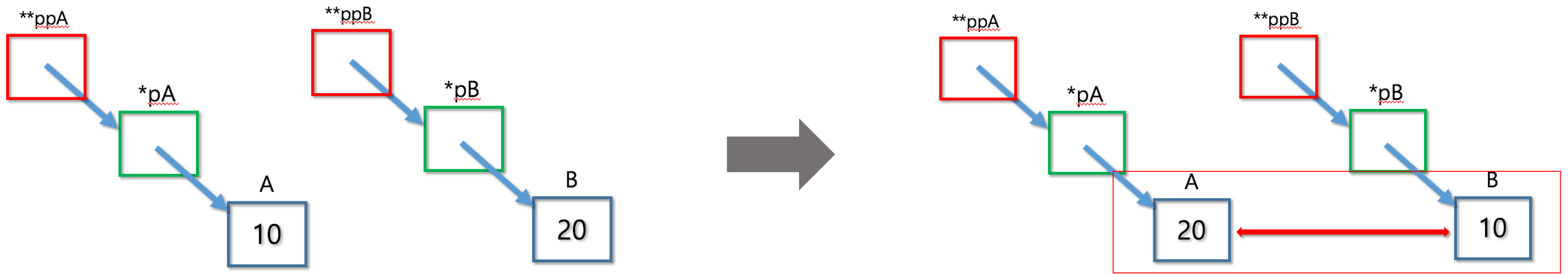
:w [name]                Save as [name]

## *Lab 2-1. Pointer for integer*

Obtain 2 integers from the standard input, and swap two integers as follows. Use pointers in your work. Print your result on the standard output.

```
>p2_1  
enter 2 integers  
2 3  
you entered  
2, 3  
after swapping  
3, 2
```

## Lab 2-1. Pointer for integer





## *Lab 2-1. Pointer for integer*

- program name : p2\_1.c
- data structure : array of integers.
- input : 2 integers separated by space.
- output : 2 integers swapped
- conditions :
  - *make a function that uses pointers to swap two numbers*

# Lab 2-1. Pointer for integer


```
#include<stdio.h>

void main(int argv, char** argc)
{
    int num1, num2;

    printf("enter 2 integers\n");
    scanf("%d%d",&num1,&num2);

    printf("you entered\n%d, %d\n",num1,num2);
    swap(&num1, &num2);
    printf("after swapping\n%d, %d\n",num1,num2);

}
```

```
void swap(int *num1, int *num2)
{
    
}
```

## *Lab 2-2. Array of characters*

Obtain a user name from the standard input, and put it on the standard output.

```
enter your name:  
Abraham Lincoln  
your name is Abraham Lincoln
```

## *Lab 2-2. Array of characters*

- program name : p2\_2.c
- data structure : array of characters.
- input : a user name(string)
- output : a user name(string)
- conditions :
  - *the length of the user name should be up to 30 characters.*
  - *blank spaces should also be part of the name (e.g. Abraham Lincoln)*

## Lab 2-3. Dynamic allocation

Use a command line argument for the total number of students ( $n > 2$ ). In the standard input, get  $n$  (the number you input) names in a single line when the instruction “enter names:” is given. Use 2D dynamic allocation to store all names in one variable. Print your result in standard output.

**arg1** **arg2**

>p2\_3 4

enter 4 names:

Bravo Charlie Delta Echo

the names you entered:

Bravo

Charlie

Delta

Echo





## *Lab 2–3. Dynamic allocation*

- Condition
  - *the number of students should be more than two.*
  - *use dynamic allocation to store names ( malloc() )*
  - *the length of the user name should be up to 30 characters.*
  - *no blank space is allowed in the name.*

## *Lab 2-3. Dynamic allocation*

### *Command line sample*

> filename arg1 arg2 arg3

int main(int argc, char \*argv[])

Filename	arg1	arg2	arg3
argv[0]	argv[1]	argv[2]	argv[3]

argc=4



## *Lab 2–3. Dynamic allocation*

- program name : p2\_3.c
- data structure : array of pointers
- input :  $n$  names
- output :  $n$  names