

CSED211 Lab 01.

Introduction & Data Lab.

18. 09. 05

Lab Introduction

Communication with TA

TJ Park Library 501

- 박채용 (Chaeyong Park)
 - HVR Lab., Building 4, 013
 - pcy8201@postech.ac.kr

TJ Park Library 503

- 박남규 (Namgyu Park)
 - HPC Lab., PIRL 441
 - skarb852@postech.ac.kr
- 조범진 (Beumjin Cho)
 - HPC Lab., PIRL 441
 - beumjincho@postech.ac.kr

About Lab Session (1/2)

- Lab session will be held
 - 37 students class @TJ Park library 501
 - 39 students class @TJ Park library 503
- Evaluation
 - Attendance: 10%
 - Please notify your absence before the lab session
 - Quiz: 10%
 - Assignment: 80%
 - Lab report: 30%
 - Source code: 50%

About Lab Session (2/2)

- We will take a brief quiz before the lab start.
 - No pre-lab report
- You have to submit your source file & the lab report for a previous week before the lab day.
 - Submit two files to the assignment board in LMS
 - File name should be student #_name.c / student #_name.doc
 - Incorrect format of files would be regarded as 0 point.
- Contents of a final report:
 1. What you did in a previous week
 2. Source code
 3. Brief explanation of your source code

Data Lab.

Bitwise Operation- Manipulating Bits

Linux Commands Basic (1/3)

- **ls:** to see a list of files/dirs
 - Options – a (all files), l (detail info.), etc.
- **cd:** change directory
 - ex) 'cd .', 'cd (~)', 'cd /subdir'
- **cp:** copy file
 - ex) 'cp a.out new.out'
- **mv:** move file
 - ex) 'mv a.out new.out', 'mv a.out ../work'
- **mkdir:** make directory
 - ex) 'mkdir homework1'

Linux Commands Basic (2/3)

- rm: remove file
 - ex) 'rm -rf /work'
- pwd: show path for current directory
- './executable file': execute a file
 - ex) './a.out'
- ↑ / ↓: previous/next command
- I + [tab]: show all commands starting with 'I'
- gcc: compile a source file to executable file
 - ex) 'gcc test.c' -> a.out , 'gcc -o output test.c' -> output

Linux Commands Basic (3/3)

- `chmod`: change mode of a file
 - `chmod +x a.out` → make a.out file executable
 - Mode is represented 3 digit binaries.
 - E.g, 755 = 111(owner) 101(group) 101(others), each bit for R/W/X
- `more`, `cat`: see contents of a file
 - ex) 'cat(or more) a.out'
- `make`: compile with pre-defined settings
- `tar`: compress & decompress files
 - ex) 'tar xvf compressed.tar' (for decompression)

* Commands in this slide will help you to do your H/W

* For more information, google 'linux commands'

Vi Command

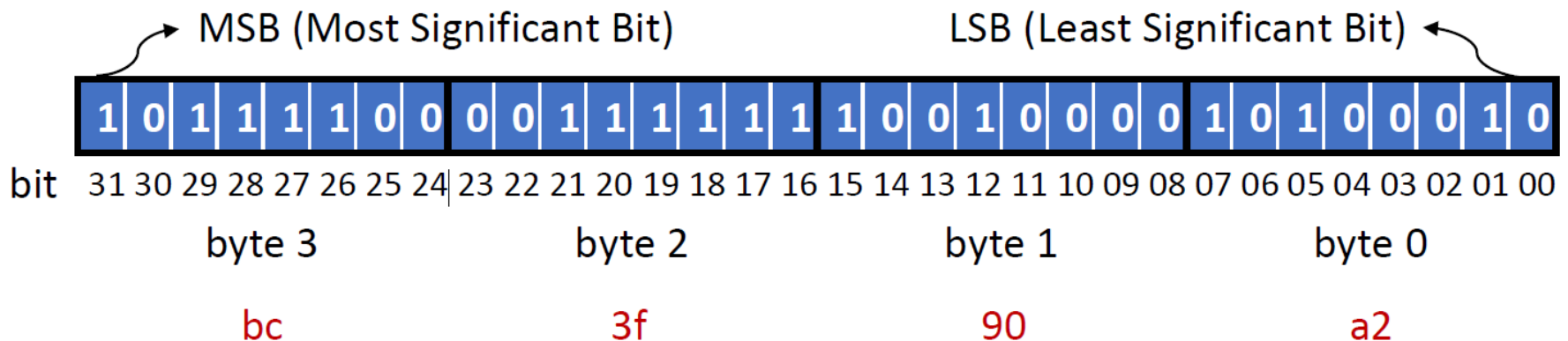
- Use vim when writing your program on Linux
- How to install?
 - `sudo apt-get update`
 - `sudo apt-get install vim`
 - (a way to install gcc would be the same)
- Usage: `vim 'your file'` -> code -> `:wq` or `ZZ`
- For the vi commands and configurations, please refer below websites
 - <https://www.cs.colostate.edu/helpdocs/vi.html>
 - <http://vimconfig.com/>

Bit and Byte

- Most systems use 4 bytes to represent an integer data type.
- 1 byte = 8 bits

- Example

- int a = 0xbc3f90a2



Bitwise operation

- In digital computer programming, a **bitwise operation** operates on one or more bit pattern or binary numerals at the level of their **individual bits**.
- Bitwise operators (\sim , $\&$, $|$, \wedge)
- Shift operators (\ll , \gg)
- Bitwise assignment operators ($\&=$, $|=$, $\wedge=$, $\ll=$, $\gg=$)

~ Operator

- Bitwise NOT operator (Complement operator)
- Invert the bits for every bit 1 the result is bit 0, and conversely for every bit 0 we have a bit 1
- Do not confuse with logical negation “!”
- Example
 - $\sim 1011 = 0100$
 - $\sim 010101011011 = 101010100100$

bit a	~a (complement of a)
0	1
1	0

& Operator

- Bitwise AND operator
- It works on the bits of the operands
- Do not confuse with logical AND “&&”

- Example

```
110011100101101101
& 100110001100000110
= 100010000100000100
```

bit a	bit b	a & b (a AND b)
0	0	0
0	1	0
1	0	0
1	1	1

| Operator

- Bitwise OR operator
- It works on the bits of the operands
- Do not confuse with logical OR “||”

- Example

```
  110011100101101101
| 100110001100000110
= 110111101101101111
```

bit a	bit b	a b (a OR b)
0	0	0
0	1	1
1	0	1
1	1	1

\wedge Operator

- Bitwise XOR (exclusive OR) operator
- The result is zero only when we have two zeroes or two ones.
- Do not confuse with logical AND “&&”

- Example

```
110011100101101101
^ 100110001100000110
= 010101101001101011
```

bit a	bit b	a ^ b (a XOR b)
0	0	0
0	1	1
1	0	1
1	1	0

<<, >> Operator

- Left shift (<<)

- 11100101 << 1 \rightarrow 11001010 ($\times 2^1$)
- 11100101 << 3 \rightarrow 00101000 ($\times 2^3$)

- Right shift (>>)

- 11100101 >> 1 \rightarrow 01110010 ($/ 2^1$)
- 11100101 >> 3 \rightarrow 00011100 ($/ 2^3$)

- Logical shift: the blanks will be filled by 0s
- Arithmetic shift: the blanks will be filled with the sign bit

Bitwise Assignment Operators

$a \&= b$	\longleftrightarrow	$a = a \& b$
$a = b$	\longleftrightarrow	$a = a b$
$a \wedge= b$	\longleftrightarrow	$a = a \wedge b$
$a \ll= b$	\longleftrightarrow	$a = a \ll b$
$a \gg= b$	\longleftrightarrow	$a = a \gg b$

- Example
 - $x \ll= 2$
 - $y |= (1 \ll 24)$
 - $a \&= b \gg 3$

Bitwise Operation Example

- Change the values of bit 24, bit 20, bit 16 to 1

Bitwise Operation Example

- Change the values of bit 24, bit 20, bit 16 to 1

```
x = 0x0;
```

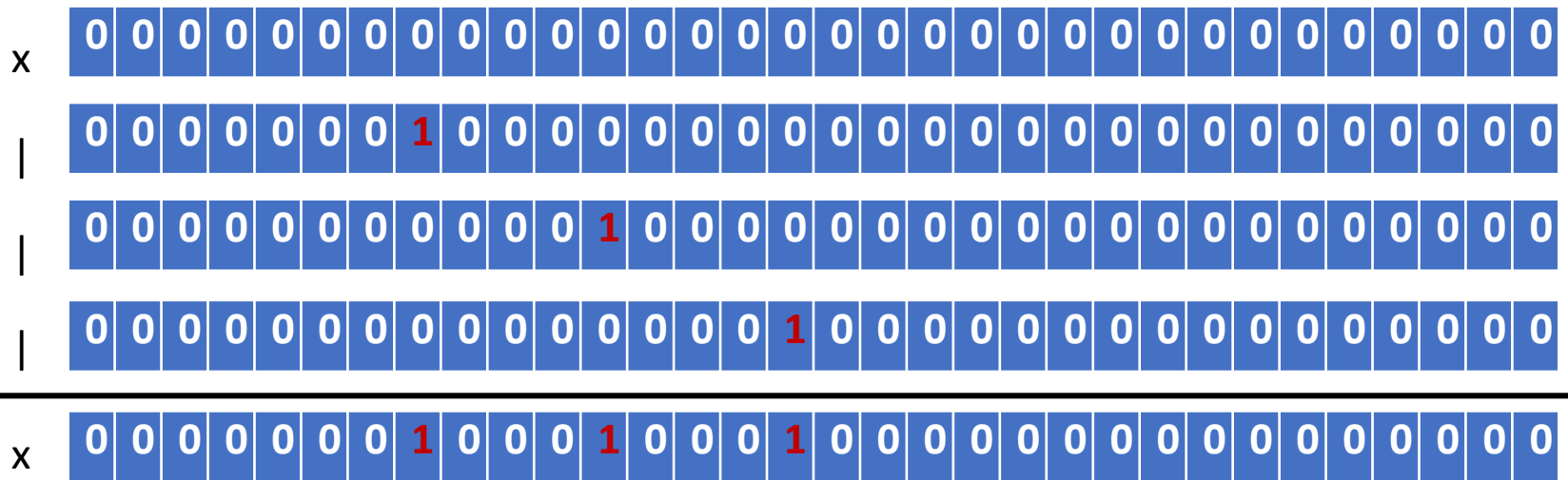
```
x |= (1<<24) | (1<<20) | (1<<16);
```

Bitwise Operation Example

- Change the values of bit 24, bit 20, bit 16 to 1

```
x = 0x0;
```

```
x |= (1<<24) | (1<<20) | (1<<16);
```



Homework

- Due: 09/12 00:00 Wed. (midnight)
- Upload your source file and report to LMS
 - Explain your answer in the final report
 - File name format(again): [student_#]_[your name].c / .doc
- Instructions

Homework Instructions (1/3)

- Use the minimum number of operator as you can
- You are allowed to use only the following:
 1. Integer constants 0 through 255(0xFF)
 2. Function arguments and local variables
 3. Unary integer operations ! ~
 4. Binary integer operations & ^ | + << >>

Homework Instructions (2/3)

- You are expressly forbidden to:
 1. Use any control constructs such as `if`, `do`, `while`, `for`, `switch`
 2. Define or use any macros
 3. Call any functions
 4. Use any other operations, such as `&&`, `||`, `-`, or `?:`
 5. Use any data type other than `int`. This implies that you cannot use arrays, structs, or unions

Homework Instructions (3/3)

- You may assume that your machine:
 1. Uses 2s complement, 32-bit representations of integers
 2. Performs right shifts arithmetically
 3. Has unpredictable behavior when shifting an integer by more than the word size

Problem1

```
/*  
 * bitOr –  $x|y$  using only  $\sim$  and  $\&$   
 * Example: bitOr(6, 3) = 7  
 * Legal ops:  $\sim$  &  
 */  
int bitOr(int x, int y) {  
    // to be implemented  
}
```

Problem2

```
/*
```

```
* addOK - Determine if can compute x+y without overflow
```

```
* Example: addOK(0x80000000,0x80000000) = 0,
```

```
*         addOK(0x80000000,0x70000000) = 1,
```

```
* Legal ops: ! ~ & ^ | + << >>
```

```
*/
```

```
int addOK(int x, int y){
```

```
// to be implemented
```

```
}
```

Problem3

```
/*  
 * negate – return -x  
 * Example: negate(1) = -1.  
 * Legal ops: ! ~ & ^ | + << >>  
 */  
int negate(int x) {  
    // to be implemented  
}
```

Problem4

```
/*
```

```
* logicalShift - shift x to the right by n, using a logical shift
```

```
* Can assume that 0 <= n <= 31
```

```
* Examples: logicalShift(0x87654321,4) = 0x08765432
```

```
* Legal ops: ~ & ^ | + << >>
```

```
*/
```

```
int logicalShift(int x, int n) {
```

```
// to be implemented
```

```
}
```

Bonus Problem

- You don't need to solve it,
but we will give you bonus points if you solve it

```
/*
```

```
* bitCount - returns count of number of 1's in word
```

```
* Examples: bitCount(5) = 2, bitCount(7) = 3
```

```
* Legal ops: ! ~ & ^ | + << >>
```

```
* Max ops: 40
```

```
*/
```

```
int bitCount(int x) {
```

```
    // to be implemented
```

```
}
```

Thank you.

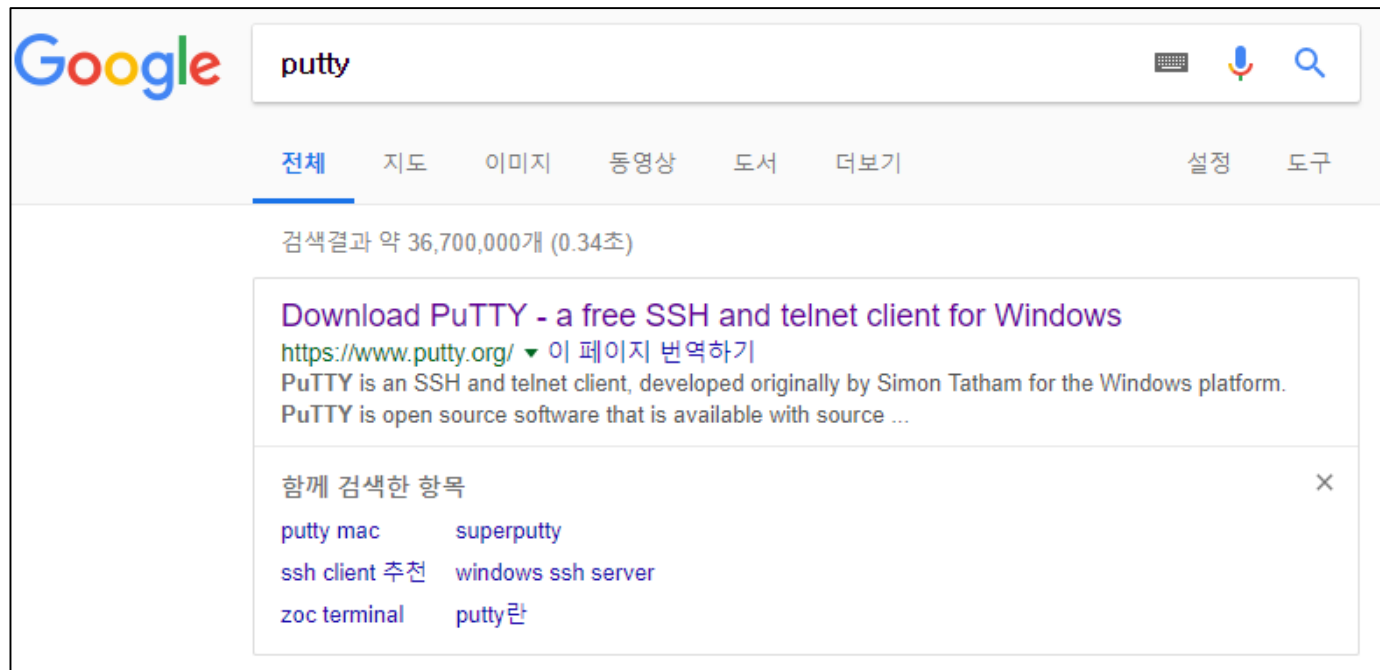
Any questions?

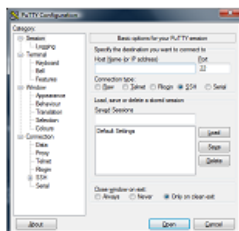
Appendix – How to access the Linux server?

You can run the code for homework-1 on Visual Studio.

However, the homework must be done on Linux machine. Thus, we will briefly explain how to install terminal programs such as PUTTY, and Xshell and access the Linux server via the programs (Xshell recommended due to its rich functionalities).

1. putty



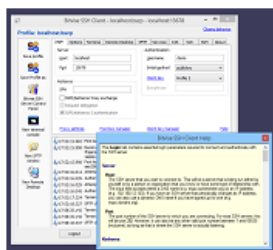


Download PuTTY

PuTTY is an SSH and telnet client, developed originally by Simon Tatham for Windows. It is available with source code and is developed and supported by a group of volunteers.

You can download PuTTY [here](#).

Below suggestions are independent of the authors of PuTTY. They are *not* to be seen as recommendations.

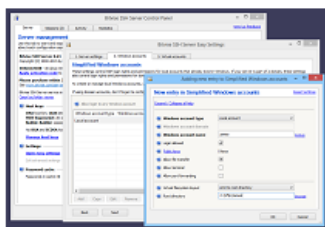


Bitvise SSH Client

Bitvise SSH Client is an SSH and SFTP client for Windows. It is developed and supported professionally, and supports all features supported by PuTTY, as well as the following:

- graphical SFTP file transfer;
- single-click Remote Desktop tunneling;
- auto-reconnecting capability;
- dynamic port forwarding through an integrated proxy;
- an FTP-to-SFTP protocol bridge.

Bitvise SSH Client is **free to use**. You can [download it here](#).



Bitvise SSH Server

Bitvise SSH Server is an SSH, SFTP and SCP server for Windows. It is robust, easy to install and use, and supports all features supported by Bitvise SSH Client, OpenSSH, and PuTTY. The SSH Server is developed and supported professionally.

You can [download Bitvise SSH Server here](#).

Alternative binary files

The installer packages above will provide all of these (except PuTTYtel), but you can do
(Not sure whether you want the 32-bit or the 64-bit version? Read the [FAQ entry](#).)

putty.exe (the SSH and Telnet client itself)


32-bit:	putty.exe	(or by FTP)	(signature)
64-bit:	putty.exe	(or by FTP)	(signature)

pscp.exe (an SCP client, i.e. command-line secure file copy)

32-bit:	pscp.exe	(or by FTP)	(signature)
64-bit:	pscp.exe	(or by FTP)	(signature)

psftp.exe (an SFTP client, i.e. general file transfer sessions much like FTP)

32-bit:	psftp.exe	(or by FTP)	(signature)
64-bit:	psftp.exe	(or by FTP)	(signature)

 PuTTY Configuration ✕

Category:

- [-] Session
 - ... Logging
- [-] Terminal
 - ... Keyboard
 - ... Bell
 - ... Features
- [-] Window
 - ... Appearance
 - ... Behaviour
 - ... Translation
 - ... Selection
 - ... Colours
- [-] Connection
 - ... Data
 - ... Proxy
 - ... Telnet
 - ... Rlogin
 - ☒ SSH
 - ... Serial

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address)	Port
programming.postech.ac.kr	2022

Connection type:

☐ Raw ☐ Telnet ☐ Rlogin ☒ SSH ☐ Serial

Load, save or delete a stored session

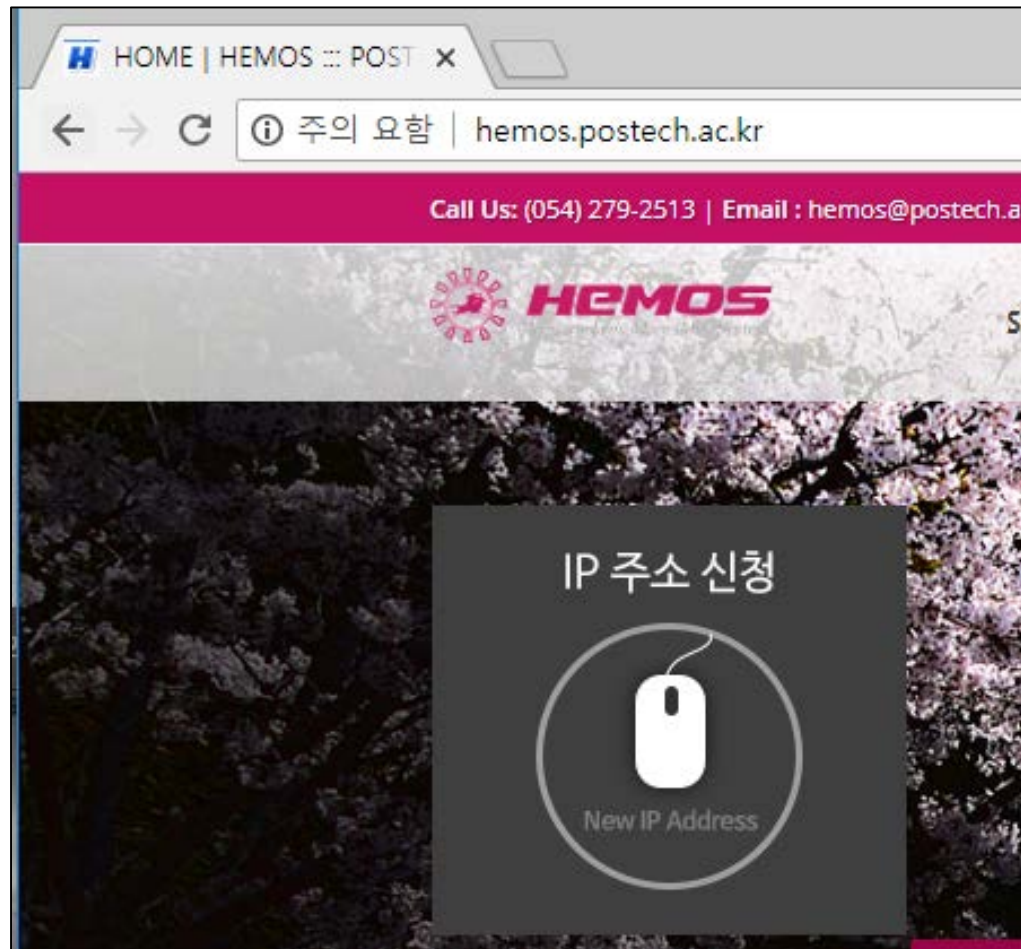
Saved Sessions

Default Settings

Close window on exit:

☐ Always ☐ Never ☒ Only on clean exit

2.Xshell



1. Login and enter the hemos server



2. Click the software library

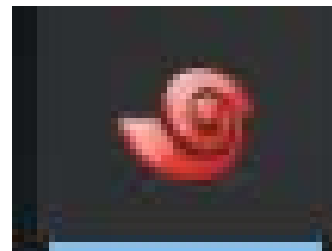
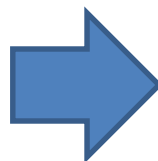
Search Package Publish Date Descending Order

Home > All Packages

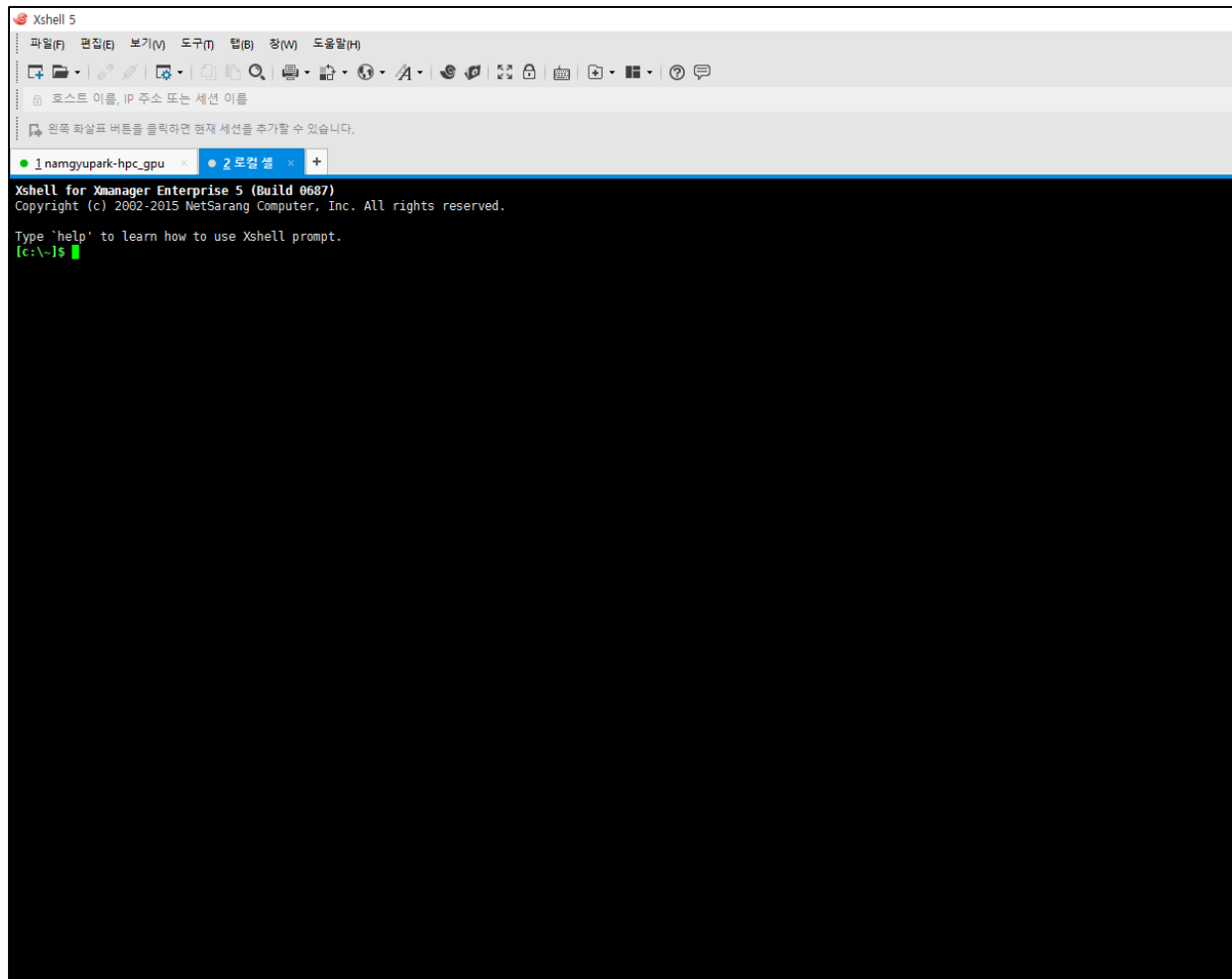
ERP (3)	GRAPHIC (1)	LANGUAGE (5)
LIBRARY (2)	OA (9)	OS (4)
PROMOTION VIDEO (2)	STATISTICS (5)	TOOL(UTILITY) (10)
VACCINE (5)	WEB (2)	



Xmanager Enterprise 5.0
51.53 MB
3507 downloads



3. Click the Tool(Utility) and download the Xmanager, then execute Xshell



4. Press the Alt + N

새 세션 등록 정보

범주(C):

- 연결
 - 사용자 인증
 - 로그인 프롬프트
 - 로그인 스크립트
 - SSH
 - 보안
 - 터널링
 - SFTP
 - TELNET
 - RLOGIN
 - SERIAL
 - 프록시
 - 연결 유지
 - 터미널
 - 키보드
 - VT 모드
 - 고급
 - 모양
 - 여백
 - 고급
 - 추적
 - 로깅
 - ZMODEM

연결

일반

이름(N): Programming_Server

프로토콜(P): SSH

호스트(H): programming.postech.ac.kr

포트 번호(P): 2022

설명(D):

다시 연결

☐ 예기치 않게 연결이 끊겼을 때 자동으로 다시 연결(A)

간격(I): 0 초 제한(L): 0 분

확인 취소

5. Enter the information of host and port number
host : programming.postech.ac.kr
Port number: 2022