

2024년 1학기 시스템프로그래밍 & 시스템 프로그래밍 실습

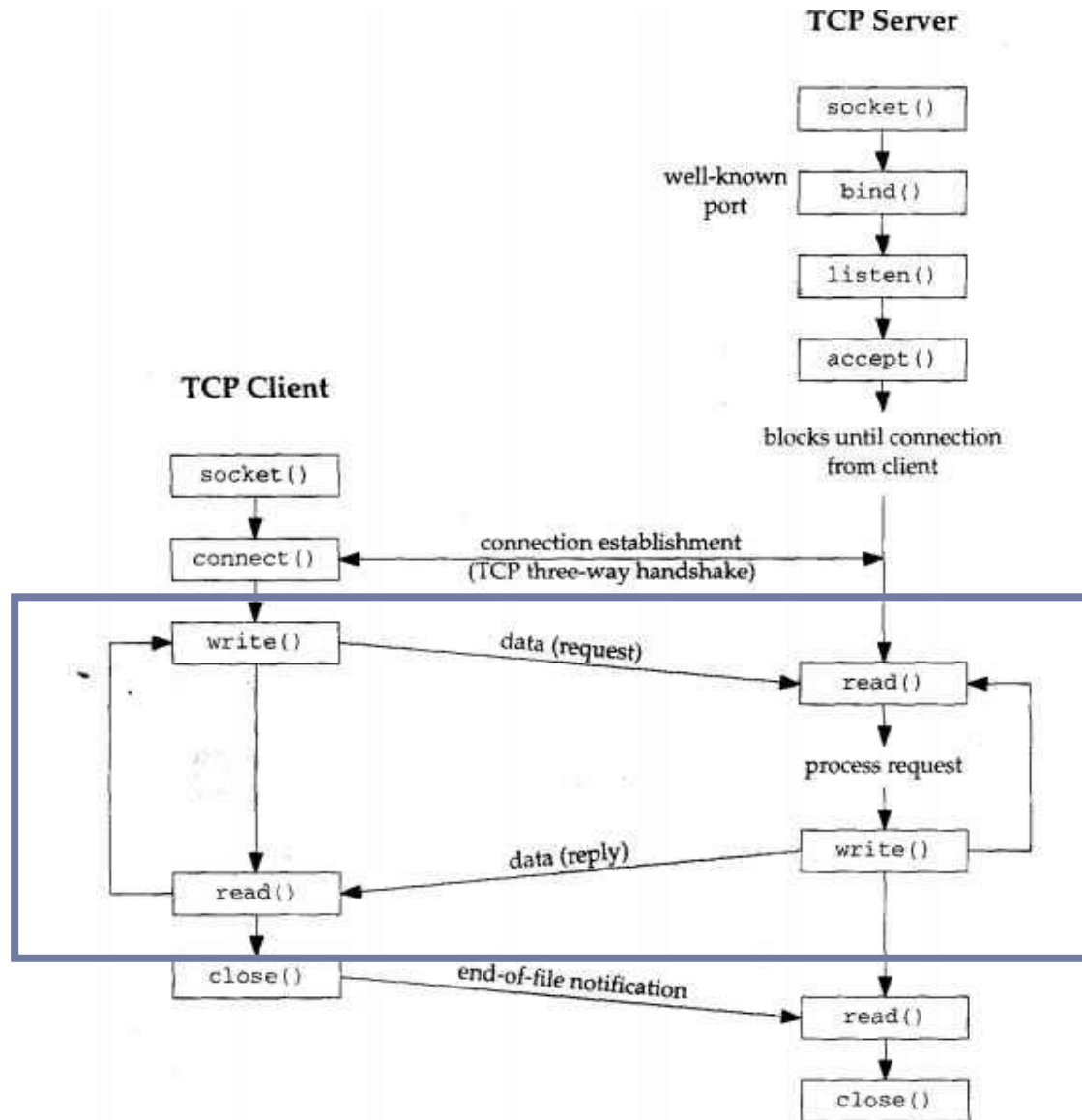
Assignment2 - 3

System Software Laboratory
College of Software and Convergence
Kwangwoon Univ.

Requirement – Socket (1/4)

- **Assignment #1 + Socket algorithm**
 - Assignment 1 combined socket algorithm.
 - Simulate socket algorithm
 - Server : socket(), bind(), listen(), accept()
 - Client : socket(), connect()

Requirement – Socket (2/4)



Requirement – Socket (3/4)

- **Server side**
 - When client connects, display client IP address , port number and child process ID
 - Display client's commands.
 - Check the FTP command and execute
 - Pass the result of execution to the client
 - Check error state.

Requirement – Socket (4/4)

- **Client side**

- Connect to server.
- Convert user command to FTP command.
- When User command is quit, convert QUIT, and quit.
- **When User push ctrl+c(SIGINT), convert QUIT, and quit.**
- Check error state.

Requirement – Command Conversion (1/2)

- **Command conversion**
 - Client side.
 - command
 - ls
 - pwd
 - dir
 - cd
 - mkdir
 - delete
 - rmdir
 - rename
 - quit
 - user command will show up in server with executed child pid
 - Ex)

```
cli[2883]
```

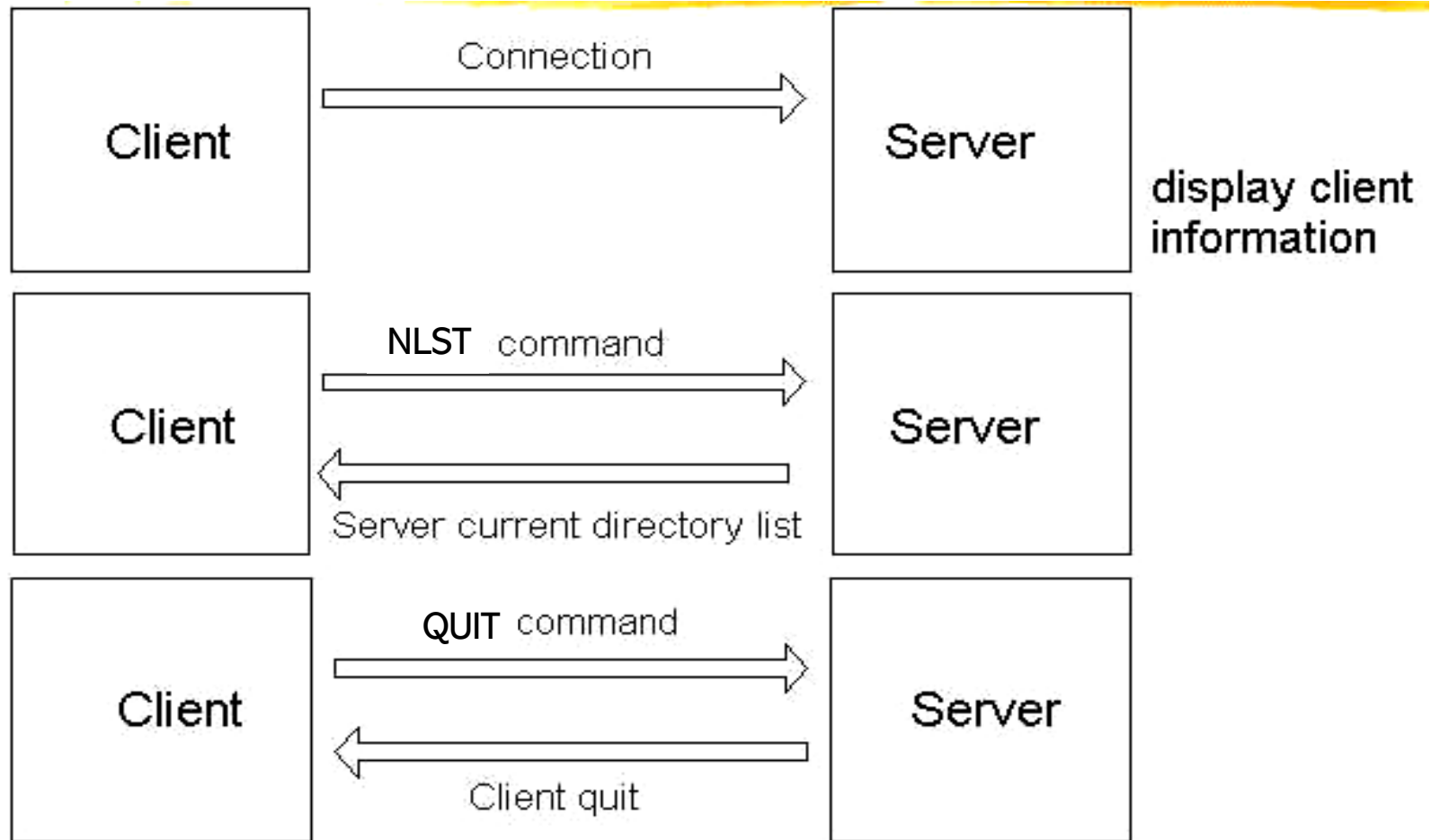
```
> ls
```

```
cli          cli.c          srv          srv.c          xxxx.pdf
```

```
srv
```

```
> NLST          [2883]
```

Requirement – Command Conversion (2/2)



Requirement – Concurrent Server (1/3)

- Concurrent Server – using `fork()`

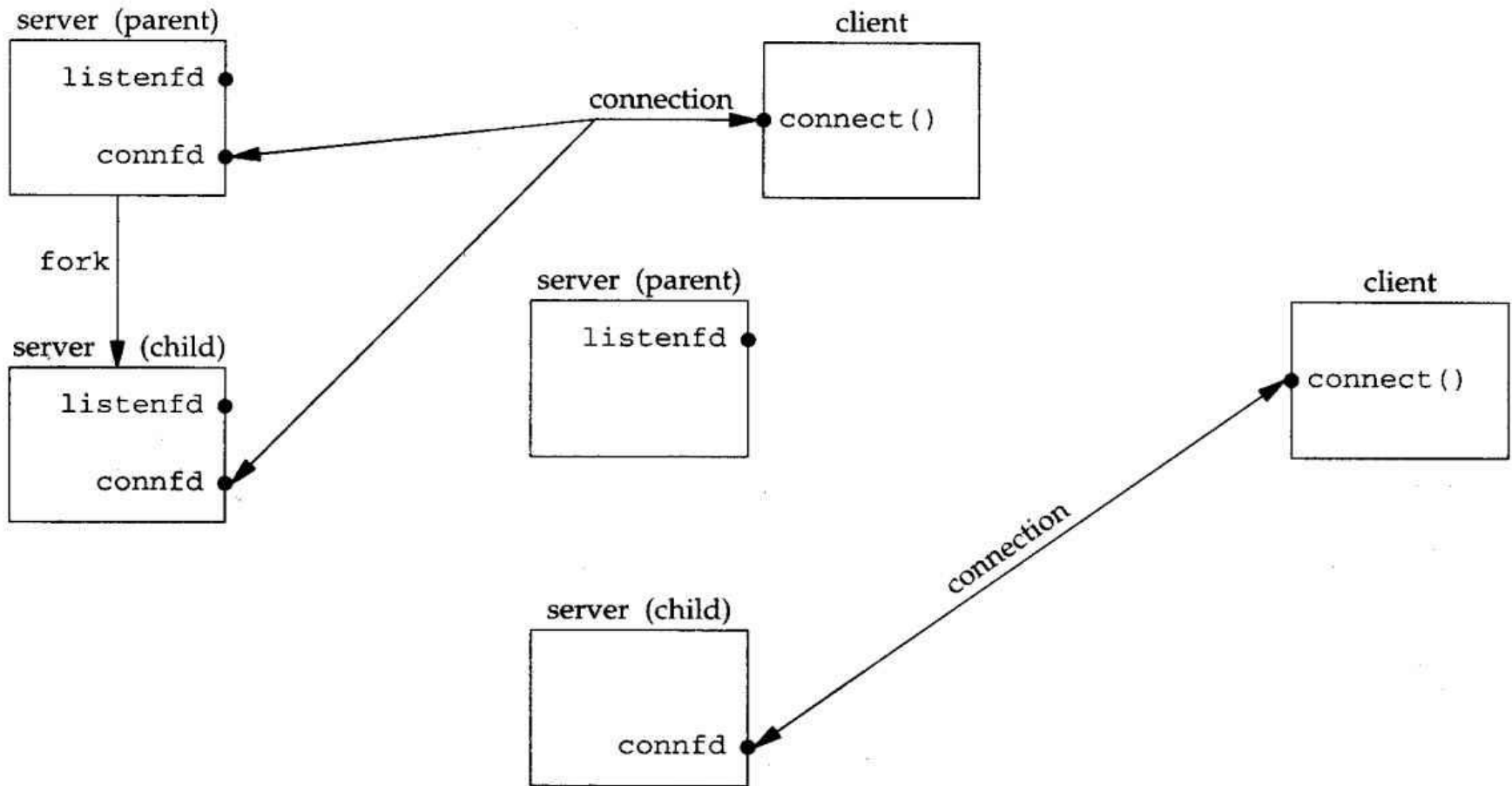
```
#include <sys/types.h>
```

```
#include <unistd.h>
```

```
pid_t fork(void);
```

- **The only way in Unix to create a new process, called once but returns twice.**
- **Returns**
 - The return value in the child is 0
 - The return value in the parent is the process ID of the new child
 - The return value on error is -1

Requirement – Concurrent Server (2/3)



Requirement – Concurrent Server (3/3)

```
pid_t pid;
int listenfd, connfd;
listenfd = socket(,,,);
bind(listenfd, ..);
listen(listenfd, LISTENQ);
for( ; ; ) {
    connfd = Accept(listenfd, ,,, );
    if( ( pid = fork() ) == 0) {
        close(listenfd);
        # 반복해서 client에서 socket을 전달받으며 해당하는
        command를 execute
        close(connfd);
        exit(0);
    }
    close(connfd);
}
```

Requirement – Signal Process

- **Signal process**

```
#include <signal.h>
```

```
void (*signal(int signo, void(*func) (int) ));
```

```
Returns : SIG_ERR if error
```

- **Use SIGCHLD, SIGALRM, SIGINT**

Requirement – Etc

▪ 다중접속 허용 & 시그널 처리

- 접속 직후, child process 의 PID 출력
- 10초 간격으로 현재 child process 의 개수와 process들의 정보를 출력
 - Process 정보 – PID, Port번호(client의 Port number), 서비스 타임(접속직후 카운트)
 - 단, 새로운 client 가 접속할 경우, 출력과 동시에 10초 interval은 그 순간부터 다시 카운트
- Client 프로그램 종료 시
 - Client side
 - quit 명령어 또는 ctrl+c를 통해 종료 후, 서버에게 QUIT를 통해 알림
 - Server side
 - QUIT 메시지를 받을 경우 해당 client와 연결된 child process 종료(delay 없음)
 - 해당 process PID를 출력
 - 해당 Process의 정보는 10초마다 출력되는 process들의 정보에서 반드시 제외
 - 해당 Process에 대한 종료 format 출력
- Server 프로그램 종료
 - Ctrl + c 로 종료.
 - Server 프로그램 종료 시, 아래 동작을 수행하는 SIGINT Handler가 동작해야 함.
 - 모든 client의 연결 종료
 - 모든 child process 종료

Requirement – Sample Result

cli#1

1. cli#1 접속
6. "quit" 로 종료

cli#2

2. cli#2 접속
5. ctrl+c 로 종료

cli#3

3. cli#3 접속 후, 명령어 대기

1. cli#1 접속 직후 서버화면

```

Client Info
client IP: 127.0.0.1
client port: 32824

Child Process ID : 2876
Current Number of Client : 1
PID  PORT  TIME
2876 32824 1
                    
```

2. cli#2 접속 직후 서버화면

```

Client Info
client IP: 127.0.0.1
client port: 32826

Child Process ID : 2878
Current Number of Client : 2
PID  PORT  TIME
2876 32824 3
2878 32826 1
                    
```

3. cli#3 접속 직후 서버화면

```

Client Info
client IP: 127.0.0.1
client port: 32827

Child Process ID : 2880
Current Number of Client : 3
PID  PORT  TIME
2876 32824 5
2878 32826 3
2880 32827 1
                    
```

4. cli#3 접속 10초 후 서버화면

```

Current Number of Client : 3
PID  PORT  TIME
2876 32824 15
2878 32826 13
2880 32827 11
Client( 2878)'s Release

Current Number of Client : 2
PID  PORT  TIME
2876 32824 25
2880 32827 21
Client( 2876)'s Release

Current Number of Client : 1
PID  PORT  TIME
2880 32827 31
                    
```

5. cli#2 종료 후 서버화면

6. cli#1 종료 후 서버화면

KWANGWOON
UNIVERSITY

13

Report Requirements

- **Ubuntu 20.04.6 Desktop 64bits 환경에서 채점**
- **Copy 발견 시 0점 처리**
- **보고서 구성**
 - **보고서 표지**
 - 수업 명, 과제 이름, 담당 교수님, 학번, 이름 필히 명시
 - 과제 이름 → Assignment2-3
 - **과제 내용**
 - Introduction
 - 과제 소개 - 4줄 이상(background 제외) 작성
 - Flow chart(4주차 강의자료 appendix 참고)
 - Pseudo code(4주차 강의자료 appendix 참고)
 - 결과화면
 - 수행한 내용을 캡처 및 설명
 - 고찰
 - 과제를 수행하면서 느낀점 작성
 - Reference
 - 과제를 수행하면서 참고한 내용을 구체적으로 기록
 - 강의자료만 이용한 경우 생략 가능

Report Requirements

▪ Softcopy Upload

- 제출 파일
 - 보고서 + 소스파일 하나의 압축 파일로 압축하여 제출(tar.gz)
 - 보고서(.pdf. 파일 변환)
 - 소스코드
 - cli.c, srv.c
 - Makefile
 - 실행파일명: cli, srv
 - 소스 코드, 실행파일명 다르게 작성 시 감점
- Tar 압축 및 해제 방법
 - 압축 시 → tar -zcvf [압축 파일명].tar.gz[폴더 명]
 - 해제 시 → tar -zxvf 파일명.tar.gz
- 보고서 및 압축 파일 명 양식
- **Assignment2_3_수강분류코드_학번**으로 작성

수강요일	이론1 월5수6	이론2 목4	실습1 금12	실습2 금56	실습3 금78
수강분류 코드	A	B	C	D	E

- 예시-이론 월5 수6 수강하는 학생인 경우
 - 보고서 Assignment2_3_A_2024123456.pdf
 - 압축 파일 명: Assignment2_3_A_2024123456.tar.gz

Report Requirements

- 실습 수업을 수강하는 학생인 경우

- 실습 과목에 과제를 제출(.tar.gz)
- 이론 과목에 간단한 .txt 파일로 제출

📄 실습수업때제출했습니다.

2022-08-29 오후 3:58

텍스트 문서

0KB

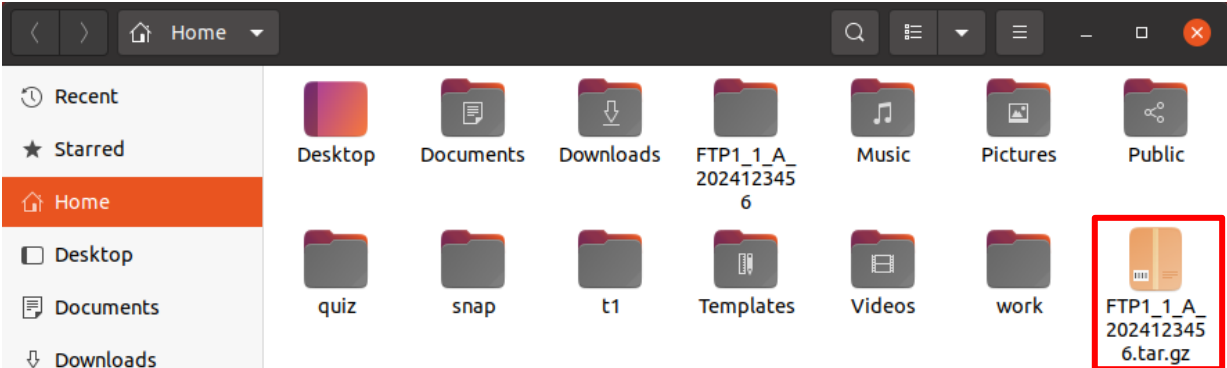
- 이론 과목에 .txt 파일 미 제출 시 감점
- .tar.gz 파일로 제출 하지 않을 시 감점

- 과제 제출

- KLAS – 강의 과제 제출
- 2024년 5월 16일 목요일 23:59까지 제출
- 딜레이 받지 않음
 - 제출 마감 시간 내 미제출시 해당 과제 **0점 처리**
 - 교내 서버 문제 발생 시, 메일로 과제 제출 허용

Appendix A. tar.gz compression

```
sslab@ubuntu: ~  
sslab@ubuntu:~$ ls  
Desktop  Downloads  Music  Public  snap  t1  Videos  
Documents FTP1_1_A_2024123456 Pictures quiz splab_commands Templates work  
sslab@ubuntu:~$ cd FTP1_1_A_2024123456/  
sslab@ubuntu:~/FTP1_1_A_2024123456$ ls  
FTP1_1_A_2024123456.pdf Makefile othercom.c  
sslab@ubuntu:~/FTP1_1_A_2024123456$ cd ..  
sslab@ubuntu:~$ tar -zcvf FTP1_1_A_2024123456.tar.gz FTP1_1_A_2024123456/  
FTP1_1_A_2024123456/  
FTP1_1_A_2024123456/othercom.c  
FTP1_1_A_2024123456/FTP1_1_A_2024123456.pdf  
FTP1_1_A_2024123456/Makefile  
sslab@ubuntu:~$ ls  
Desktop  FTP1_1_A_2024123456  Pictures  snap  Templates  
Documents FTP1_1_A_2024123456.tar.gz Public splab_commands Videos  
Downloads Music quiz t1 work  
sslab@ubuntu:~$
```



The screenshot shows a file manager window with a sidebar on the left containing 'Recent', 'Starred', 'Home', 'Desktop', 'Documents', and 'Downloads'. The main area displays a grid of folders and files. A red box highlights the file 'FTP1_1_A_2024123456.tar.gz' in the bottom right corner of the grid.

Appendix B. Comment 작성 요령 (1/3)

- File Head Comment

```
////////////////////////////////////  
// File Name      : Main.c                               //  
// Date          : 2024/03/01                             //  
// OS            : Ubuntu 20.04.6 LTS 64bits              //  
//              //  
// Author        : Hong Gil Dong                         //  
// Student ID    : 2024123456                             //  
// ----- //  
// Title : System Programming Assignment #1-1 ( ftp server ) //  
// Description : ...                                       //  
////////////////////////////////////
```

Appendix B. Comment 작성 요령 (2/3)

- Function Head Comment

```
////////////////////////////////////  
// InsertNode                                                    //  
// =====                                                    //  
// Input: Node* -> Insert Node,                                  //  
//           Node* -> Column node before insert node          //  
//           Node* -> Row node before insert node              //  
//           (Input parameter Description)                       //  
// Output: int  - 1 success                                       //  
//           0 fail                                              //  
//           (Out parameter Description)                         //  
// Purpose: Inserting node                                       //  
////////////////////////////////////
```

Appendix B. Comment 작성 요령 (3/3)

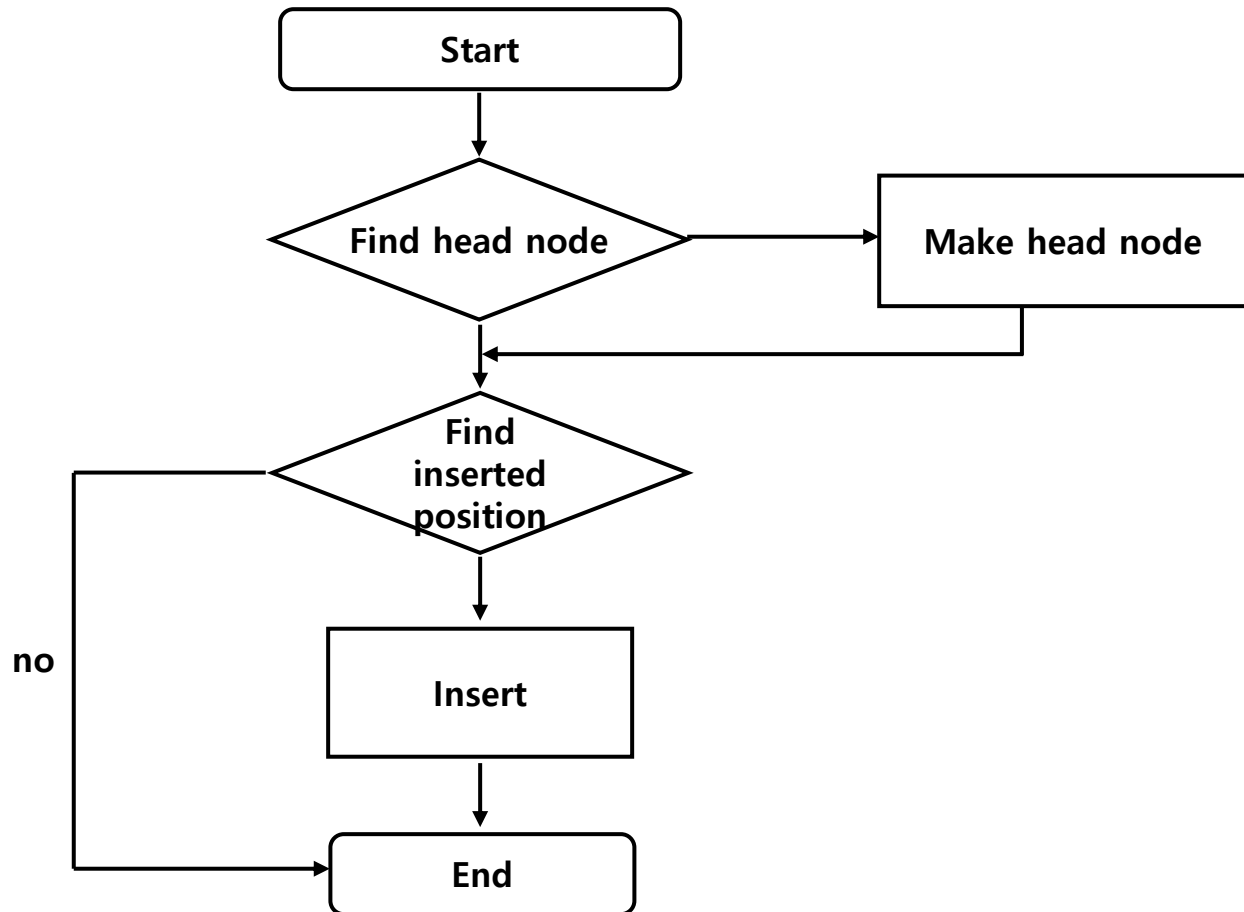
- In-line Comment

```
//////////////////////////////////// Row insert //////////////////////////////////////
if( pRowPos->pNextRow != pRowPos ) {
    pTemp->pNextRow = pRowPos->pNextRow;          // pTemp set next row
    if( !( pRowPos->pNextRow->bHead ) ){
        pRowPos->pNextRow->NodeItem.pPrevRow = pTemp;
    } // end of if
} // end of if
else {
    pTemp->pNextRow = pRowPos;                    // pTemp set next row
} // end of else
pTemp->NodeItem.pPrevRow = pRowPos;              // pTemp set previous row
pRowPos->pNextRow = pTemp;
//////////////////////////////////// End of row insert //////////////////////////////////////
```

Appendix C. 보고서 작성 요령 (1/2)

- Algorithm – Flow Chart (Each function)

- E.g.



Appendix C. 보고서 작성 요령 (2/2)

- Algorithm – Pseudo Code

```
FixHeap(Node *root, Key k)
{
    Node vacant, largerChild;
    vacant = root;
    while( vacant is not leaf ) {
        largerChild = the child of vacant with the larger key;
        if( k < largerChild's Key ) {
            copy largerChild's key to vacant;
            vacant = largerChild;
        }
        else exit loop;
    }
}
```