

2019년 1학기 시스템프로그래밍실습 4주차


# ls basics

**System Software Laboratory**  
College of Software and Convergence  
Kwangwoon Univ.

# Contents

- **gcc**
  - Example(binary execution)
- **ls – list directory contents**
- **Experiment (simple “ls” implementation)**
  - Data types
  - System Calls & Functions
  - Results
- **Assignment**

# gcc

- **GNU Compiler Collection**
- **Options**
  - `c` : compile or assemble. but do not link
  - `o file` : Place output in file
    - If not specified, *a.out* by default
- **Example**
  - `gcc -c file.c`  
→ file.o generated
  - `gcc -o binary file1.c file2.c`   
→ executable file (i.e. *binary*) generated

# Binary execution

- **Whenever execute program, you must specify the path**

- . : current directory
- .. : parent directory

- **You can execute program like this**

- ./binary
- e.g. "test.c"

```
#include <stdio.h>

int main() {
    printf("Binary exectuion\n");
    return 0;
}
```

```
sp2018110609@ubuntu:~/test$ ls
test.c
sp2018110609@ubuntu:~/test$ gcc -c test.c
sp2018110609@ubuntu:~/test$ ls
test.c test.o
sp2018110609@ubuntu:~/test$ gcc -o run test.o
sp2018110609@ubuntu:~/test$ ls
run test.c test.o
sp2018110609@ubuntu:~/test$ ./run
Binary exectuion
sp2018110609@ubuntu:~/test$
```

# ls – list directory contents

- **Synopsis**

- `$ ls [OPTION]... [FILE]...`

- **Description**

- List information about the FILEs (current directory by default)

- **Options**

- `-a`
    - do not ignore entries starting with `.` (i.e. *hidden file or directory*)
  - `-l`
    - use a long listing format
    - File mode, number of links, owner name, group name, number of bytes in the file, abbreviated month, day-of-month file was last modified, hour file last modified, minute file last modified, and the pathname

# ls – list directory contents (cont'd)

## ► Example

```
sp2018110609@ubuntu: ~  
sp2018110609@ubuntu:~$ ls -al  
total 120  
drwxr-xr-x 17 sp2018110609 sp2018110609 4096 Mar 17 18:32 .  
drwxr-xr-x  4 root          root          4096 Mar 11 19:12 ..  
-rw----- 1 sp2018110609 sp2018110609   56 Mar 11 22:09 .bash_history  
-rw-r--r-- 1 sp2018110609 sp2018110609  220 Mar 11 19:12 .bash_logout  
-rw-r--r-- 1 sp2018110609 sp2018110609 3771 Mar 11 19:12 .bashrc  
drwx----- 11 sp2018110609 sp2018110609 4096 Mar 17 18:21 .cache  
drwx----- 14 sp2018110609 sp2018110609 4096 Mar 17 18:27 .config  
drwxr-xr-x  2 sp2018110609 sp2018110609 4096 Mar 11 22:08 Desktop  
-rw-r--r-- 1 sp2018110609 sp2018110609   25 Mar 11 22:08 .dmrc  
drwxr-xr-x  2 sp2018110609 sp2018110609 4096 Mar 11 22:08 Documents  
drwxr-xr-x  2 sp2018110609 sp2018110609 4096 Mar 11 22:08 Downloads  
-rw-r--r-- 1 sp2018110609 sp2018110609 8980 Mar 11 19:12 examples.desktop  
drwx-----  2 sp2018110609 sp2018110609 4096 Mar 11 22:09 .gconf  
drwx-----  3 sp2018110609 sp2018110609 4096 Mar 17 18:20 .gnupg  
-rw----- 1 sp2018110609 sp2018110609   636 Mar 17 18:20 .ICEauthority  
drwx-----  3 sp2018110609 sp2018110609 4096 Mar 11 22:08 .local  
drwxrwxr-x  2 sp2018110609 sp2018110609 4096 Mar 17 18:30 ls  
drwxr-xr-x  2 sp2018110609 sp2018110609 4096 Mar 11 22:08 Music  
drwxr-xr-x  2 sp2018110609 sp2018110609 4096 Mar 11 22:08 Pictures  
-rw-r--r-- 1 sp2018110609 sp2018110609   655 Mar 11 19:12 .profile  
drwxr-xr-x  2 sp2018110609 sp2018110609 4096 Mar 11 22:08 Public  
drwxr-xr-x  2 sp2018110609 sp2018110609 4096 Mar 11 22:08 Templates  
drwxrwxr-x  2 sp2018110609 sp2018110609 4096 Mar 17 18:32 test  
drwxr-xr-x  2 sp2018110609 sp2018110609 4096 Mar 11 22:08 Videos  
-rw----- 1 sp2018110609 sp2018110609 1079 Mar 17 18:32 .viminfo  
-rw----- 1 sp2018110609 sp2018110609   102 Mar 17 18:20 .Xauthority  
-rw----- 1 sp2018110609 sp2018110609    82 Mar 17 18:20 .xsession-errors  
-rw----- 1 sp2018110609 sp2018110609 1098 Mar 11 22:09 .xsession-errors.old  
sp2018110609@ubuntu:~$
```

# ls – list directory contents (cont'd)

- **Data types**
  - `typedef struct __dirstream DIR`
  - `struct dirent`
  - `struct passwd`
  - `struct stat`
  - `struct tm`
- **System Calls & Functions**
  - `opendir(), readdir(), closedir()`
  - `stat()`
  - `getgrgid(), getpwuid()`
  - `localtime()`
  - `getwd()`
  - `getopt()`
  - `fnmatch()`

# Data types

- **header : <dirent.h>**
- **Data type : typedef struct \_\_dirstream DIR**
  - The DIR data type represents a directory stream



# Data types (cont'd)

- **header** : <dirent.h>
- **Data type** : struct dirent
- **Members** :
  - `__ino_t`            `d_ino`                    // inode number
  - `char d_name[256]`            // filename

On Linux, the *dirent* structure is defined as follows:

```
struct dirent {
    ino_t      d_ino;      /* inode number */
    off_t      d_off;      /* offset to the next dirent */
    unsigned short d_reclen; /* length of this record */
    unsigned char d_type;   /* type of file; not supported
                           by all file system types */
    char       d_name[256]; /* filename */
};
```

Source : [https://linux.die.net/man/3/readdir\\_r](https://linux.die.net/man/3/readdir_r)

# Reading directories

- `#include <dirent.h>`

**DIR \*opendir(const char \*name);**

- opens a directory stream corresponding to the directory **name**
- returns a pointer to the directory stream (on error, NULL is returned)
- The stream is positioned at the first entry in the directory

# Reading directories (cont'd)

- **#include <dirent.h>**

**struct dirent \*readdir(DIR \*dirp);**

- returns a pointer to a dirent structure representing the next directory entry in the directory stream pointed to by **dirp**
- It returns NULL on reaching the end of the directory stream or if an error occurred.
- If an error occurs, NULL is returned and **errno** is set appropriately.

# Reading directories (cont'd)

- `#include <dirent.h>`

`int closedir(DIR *dirp);` 

- closes the directory stream associated with `dirp`
- The directory stream descriptor `dirp` is not available after this call.
- returns 0 on success. On error, -1 is returned

# 실습

## 소스 코드

```
#include <stdio.h>
#include <dirent.h>

int main(){
    DIR *dirp;
    struct dirent *dir;

    _____ = opendir(".");

    while(( _____ )!=NULL){
        printf("%s\n", _____ );
    }

    closedir(_____);

    return 0;
}
```

\$ vi ls.c

## 결과 화면

```
sp2018110609@ubuntu:~/ls$ ls
ls.c  spls
sp2018110609@ubuntu:~/ls$ ls -a
.  ..  ls.c  spls
sp2018110609@ubuntu:~/ls$ ./spls
..
spls
ls.c
.
```

\$ ./spls

# 1차 퀴즈

- 날짜

- 2019/03/30(Sat)

- 시간

- 오전 11:00 ~ 12:00
- 단, 11시 30분 이후 입실 불가

- 장소

- 첨부 파일에 있는 내용 확인 후, 학번에 맞게 비마관 211호,비마관 401-1호 퀴즈 응시

- 내용

- 과제: 1차 과제
- 강의자료
  - 1주차
  - 2주차
  - 3주차

2019년 1학기 시스템프로그래밍실습 4주차

# Assignment 2-1

**System Software Laboratory**  
College of Software and Convergence  
Kwangwoon Univ.

# Assignment 2-1

- To do List
  - 파일 이름만 출력하는 Simple ls 구현
    - 코드를 구현하고 프로그램 실행결과 캡처

```
sslab@ubuntu:~/work$ ls
ass      exam      file3.txt  html      LINUX      test      text.txt
empty.txt file2.txt  hello.c    html_ls.html simple_ls  testing
sslab@ubuntu:~/work$ ./simple_ls
ass
empty.txt
exam
file2.txt
file3.txt
hello.c
html
html_ls.html
LINUX
simple_ls
test
testing
text.txt
sslab@ubuntu:~/work$ ./simple_ls html
ex.html
file1.txt
file2.txt
file3.txt
hello_copy.txt
sslab@ubuntu:~/work$
```



# Assignment 2-1 Requirements

## ■ Code Requirements

- 하나의 **파일 경로(디렉토리)**의 안의 파일 이름들을 출력
- 파일경로를 인자로 주지 않을 경우 default로 current directory 결과 출력
- 히든 파일(starting with .)은 출력하지 않음
- 파일 이름은 정렬되어야 함
  - 알파벳 순으로 정렬(abc)
  - 대소문자 구분 없이 정렬(aaa, Abc, ada)
  - **정렬 함수 라이브러리 사용 금지**
- 다음 페이지의 “예외 처리”를 필히 반영
- 반드시 **본 자료의 Appendix에 명시된 형태로 소스코드에 주석을 포함해야 함**

## ■ Makefile Requirements

- 실행 파일이 “simple\_ls”로 생성되도록 Makefile 작성
- 컴파일 도중 warning 발생 시 감점
- “\$ make” 를 통해 실행 파일이 생성 되지 않거나 문제가 발생하는 경우, 0점

# Assignment 2-1 Requirements (cont'd)

## 예외 처리

- 디렉토리가 아닌 파일을 입력하는 경우
- 존재하지 않는 디렉토리를 입력하는 경우
- 두 개 이상의 파일 경로를 입력하는 경우

```
sslab@ubuntu:~/work$ ls
empty.txt  file3.txt  hello.c  LINUX      testing
file2.txt  file.txt   html     simple ls  text.txt
sslab@ubuntu:~/work$ ./simple ls hello.c
simple_ls: cannot access 'hello.c' : No such directory
sslab@ubuntu:~/work$ ./simple_ls Not_file
simple_ls: cannot access 'Not_file' : No such directory
sslab@ubuntu:~/work$ ./simple_ls html Linux
simple_ls: only one directory path can be processed
sslab@ubuntu:~/work$
```

# Report Requirements

## ■ 표지

- 다음의 내용은 **필히** 기록
  - 과제 이름 (e.g. Assignment#2-1)
  - 분반 (요일, 담당 교수님)
  - 본인 인적 사항 (학번, 이름)

## ■ 과제 내용

- Introduction : 5줄 이하
- Flow chart : 본 자료의 Appendix 참고
- Pseudo code : 본 자료의 Appendix 참고
- Result : 수행한 내용을 캡처 이미지와 함께 설명
- Conclusion : 결론 및 고찰

## ■ 보고서 이름은 “실습 요일\_과제명\_학번”으로 수정

- e.g. 월1,2 → **mon\_2-1\_2017202000.pdf**
- e.g. 화3,4 → **tue\_2-1\_2017202000.pdf**
- e.g. 금5,6 → **fri\_2-1\_2017202000.pdf**

# Assignment 2-1

## ▪ Softcopy Upload

- 제출 파일
  - 보고서
  - simple\_ls.c, Makefile
- 위 파일들을 압축해서 제출 (파일명: 실습 요일\_2-1\_학번.tar.gz)
  - e.g. 월1,2 → **mon\_2-1\_2017202000.tar.gz**
  - e.g. 화3,4 → **tue\_2-1\_2017202000.tar.gz**
  - e.g. 금5,6 → **fri\_2-1\_2017202000.tar.gz**
- U-Campus의 과제 제출에 **4월 5일(금) 23:59:59까지** 제출
  - U-Campus에 올린 후 다시 다운로드 받아서 **파일이 정확한지 확인**
- 미리 공지한 바와 같이, **delay 받지 않음 (예외 없음)**
- **Ubuntu 16.04 64bits 환경**에서 채점

## ▪ 과제 질문 관련

- 해당 과제 출제 담당 조교에게 이메일로 문의 → **남건욱 조교** ([ngotic@kw.ac.kr](mailto:ngotic@kw.ac.kr))
- 과제 제출 마감 당일에는 **오후 4시까지** 도착한 질문 메일에만 답변

2019년 1학기 시스템프로그래밍실습 4주차

# Appendix

**System Software Laboratory**  
College of Software and Convergence  
Kwangwoon Univ.

# Comment 작성 요령

- File Head Comment

```
////////////////////////////////////  
// File Name      : Main.c                               //  
// Date          : 2018/03/01                             //  
// Os            : Ubuntu 16.04 LTS 64bits                //  
// Author        : Hong Gil Dong                         //  
// Student ID    : 2018123456                             //  
// ----- //   
// Title : System Programming Assignment #1-1 (proxy server) //  
// Description : ...                                       //  
////////////////////////////////////
```

# Comment 작성 요령 (cont'd)

- 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                                     //  
////////////////////////////////////
```

# Comment 작성 요령 (cont'd)

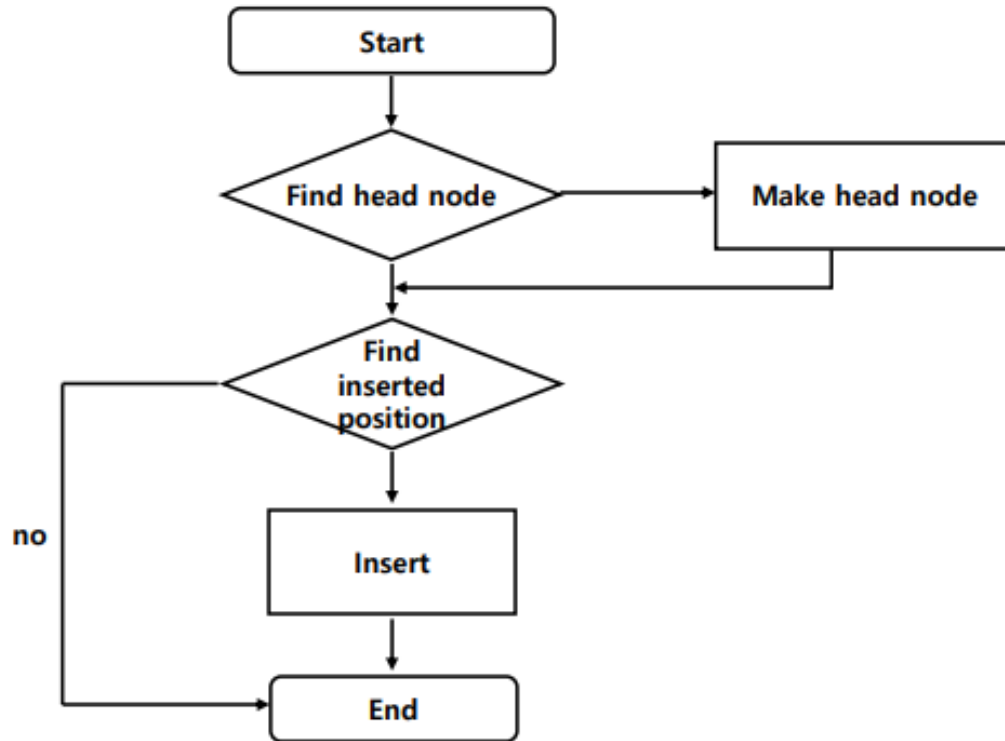
- In-line Comment

```
//////////////////// 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 //////////////////////
```



# Flow Chart 작성 요령

- Algorithm – Flow Chart (Each function)



# Pseudo Code 작성 요령

- 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;
    }
}
```