Lecture 10

표준 라이브러리

리뷰

- 라이브러리
  - 링크: 심볼을 라이브러리에서 제공된 전역 변수나 함수들과 연결하고 메모리 번지를 할당하는 것
  - 전적링크: 프로그램 컴파일 때 링크를 실행하는 것(전적 라이브러리)
  - 동적링크: 프로그램 실행 때 링크를 실행하는 것(공유 라이브러리)
  - 공유 라이브러리
    - 확장자: `.so`(Linux), `.dll`(Windows)
    - 표준 라이브러리 경로에 없을 경우, 라이브러리 경로를 지정해 야 함
  - 라이브러리 컴파일

    `gcc -c mylib.c -o mylib.o`

    `ar rcs libmylib.a mylib.o` (전적 라이브러리)

    `gcc -shared -fPIC -o libmylib.so mylib.o` (공유 라이브러리, Linux)

    `gcc -shared -fPIC -o libmylib.dll mylib.o` (공유 라이브러리, Windows)

리뷰

- 힙(heap)
  - 최댓값이나 최솟값을 빠르게 찾아내도록 만들어진 자료구조
  - 최대 힙(max heap)
    - 부모 노드의 값이 자식 노드의 값보다 크거나 같음
  - 최소 힙(min heap)
    - 보모 노드의 값이 자식 노드의 값보다 작거나 같음
  - 대표적인 연산
    - `heapify()` : 힙의 특징 유지
    - `build_heap()` : 힙 만들기
    - `Insert()` : 힙에 데이터 추가
    - `extract()` : 힙에서 최댓(최솟)값을 제거하여 돌려보냄
    - `increase_key()` : 노드의 키 값을 증가시킴

리뷰

- B-Trees(balanced search trees)
  - 하드디스크 접속 횟수를 최소화하기 위해 만들어진 자료구조
  - 데이터베이스, 파일 시스템 등에 많이 쓰임
  - Root 노드는 메인 메모리에서, 자식 노드는 하드디스크에서 있으며, 필요시 root 노드를 통해 메인 메모리로 불려올 수 있음
  - 대표적인 연산
    - 탐색
    - 빈 트린 만들기
    - 트리에 키 추가
    - 트리 노드 분할
    - 트리에서 키 삭제

<stdio.h>: 파일 열기/닫기

- `FILE *fopen(const char *filename, const char *mode)`
  - 파일을 열어 stream [1]으로 읽을 수 있게 해줌
  - 파일이 열리지 않으면 `NULL`이 리턴됨
  - `mode` 매개변수
    - `r` 파일 읽기
    - `w` 파일 쓰기, 원래 있었던 내용이 없어짐
    - `a` 파일 덧붙여 쓰기
    - `r+` 파일 수정하기(읽고 쓸 수 있음)
    - `w+` 파일 수정을 위해 파일 만들기, 원래 있었던 내용이 없어짐
    - `a+` 파일 수정을 위해 파일 열거나 만들기, 파일 끝에 덧붙여 쓰게 됨

[1] stream은 디스크나 그 외의 장치를 오가는 데이터의 모임을 의미함

<stdio.h>: 파일 열기/닫기

- `FILE *freopen(const char *filename, const char *mode, FILE *stream)`
  - 파일을 `mode`에 맞게 열고 `stream`을 리턴함
  - 에러 시 `NULL`이 리턴됨
  - 보통 `stdin`, `stdout` 또는 `stderr`와 관련된 파일을 변경하는 데 사용됨
- `int fflush(FILE *stream)`
  - 출력 stream에 사용하고, stream이 버퍼에 저장되게 해줌
  - 입력 stream에 사용하는 것은 정의되어 있지 않음
  - 에러 시 `EOF`가 리턴됨
- `int fclose(FILE *stream)`
  - 사용되던 버퍼를 지우고, `stream`을 닫음
  - 에러 시 `EOF`가 리턴됨

<stdio.h>: 파일 조작

- `int remove(const char *filename)`
  - 파일을 지우기
  - 에러 시 0이 아닌 값을 리턴함
- `int rename(const char *oldname, const char *newname)`
  - 파일의 이름을 바꾸기
  - 에러 시 0이 아닌 값을 리턴함

<stdio.h>: 임시 파일

- `FILE *tmpfile(void)`
  - `"wb+" mode`인 임시 파일을 만들어 줌
  - `fclose`를 호출하거나 프로그램이 끝날 때 임시 파일이 자동 없어짐
  - 임시 파일을 만들어 줄 수 없으면 `NULL`가 리턴됨
- `char *tmpnam(char s[L_tmpnam])`
  - 존재하는 파일의 이름이 아닌 새로운 이름을 만들어줌
  - 호출할 때마다 다른 이름을 만들어줌
  - 결과를 리턴하는 방법
    - `tmpnam(NULL)`으로 호출될 경우 문자열의 포인터를 리턴해줌
    - `tmpnam(s)`으로 호출될 경우 `s`에 저장해줌

<stdio.h>: 직접 입출력

- `size_t fread(void *ptr, size_t size, size_t nobj, FILE *stream)`
  - `stream`에서 읽은 입력을 ptr이라는 배열에 넣음
  - 크기는 `size`로 지정하고 최대 대상체의 개수는 `nobj`로 정해줌
  - 읽은 대상체의 개수를 리턴함
  - 입력 상태를 나타내기 위해서 `feof`와 `ferror`를 사용해야 함

- `size_t fwrite(const void *ptr, size_t size, size_t nobj, FILE *stream)`
  - 배열 `ptr`의 내용을 `stream`에 씀
  - 크기는 `size`로 지정하고 대상체의 개수는 `nobj`로 정해줌
  - 쓰인 대상체의 개수를 리턴하며, 이 개수가 `nobj`보다 작으면 에러임

<stdio.h>: 파일 배치

- `int fseek(FILE *stream, long offset, int origin)`
  - `stream`의 위치를 선정하고, 다음의 읽거나 쓰기는 선정된 위치에 함
  - Binary 파일인 경우 번지는 시작 번지 `origin`과 오프셋 번지 `offset`에 의해 정해짐
    - `SEEK_SET` : 시작
    - `SEEK_CUR` : 현재 위치
    - `SEEK_END` : 파일 끝
  - 텍스트 stream인 경우 시작 번지는 `SEEK_SET`으로 되고, `offset`는 `0`이 되든지 `ftell`에 의해 정해야 함
  - 에러 시 `0`이 아닌 값을 리턴함
- `long ftell(FILE *stream)`
  - 현재의 파일 위치를 리턴하고, 에러 시 -1L을 리턴함

<stdio.h>: 파일 배치

- `int rewind(FILE *stream)`
  - 파일 위치는 시작으로 선정됨
  - `fseek(stream, 0L, SEEK_SET)`과 같음
- `int fgetpos(FILE *stream, fpos_t *ptr)`
  - `stream`의 현재 위치를 `*ptr`에 넣어줌
  - `fpos_t`의 형은 알맞은 형태가 됨
  - 에러 시 `0`이 아닌 값을 리턴함
- `int fsetpos(FILE *stream, const fpos_t *ptr)`
  - `stream`을 `*ptr`이 지정하는 위치에 놓음
  - 에러 시 `0`이 아닌 값을 리턴함

<stdio.h>: 파일 에러

- `void clearerr(FILE *stream)`
  - `stream` 처리 중 방생한 에러 신호나 파일 끝을 알리는 시호를 지움
- `int feof(FILE *stream)`
  - `stream`의 처리 중 파일의 끝을 알리는 신호가 들어오면 `0`이 아닌 값을 리터함
- `int ferror(FILE *stream)`
  - `stream` 처리 중 에러 신호가 생기면 `0`이 아닌 값을 리턴함

<ctype.h>: 문자 분류

`isalnum(c)`	`isalpha(c)` 또는 `isdigit(c)`가 만족되는 경우
`isalpha(c)`	`isupper(c)` 또는 `islower(c)`가 만족되는 경우
`isupper(c)`	영문 대문자
`islower(c)`	영문 소문자
`isdigit(c)`	10진수
`iscntrl(c)`	제어 문자인 경우
`isgraph(c)`	출력되는 문자(공백 제외)
`isprint(c)`	출력되는 문자(공백 포함)
`ispunct(c)`	영문자, 숫자, 공백이 아닌 문자
`isspace(c)`	공백, 행바꿈, 케리지 리턴, 탭, 페이지 바꿈 등
`isxdigit(c)`	16진수

<string.h>: 메모리 함수

- `void *memcpy(void *dst, const void *src, size_t n)`
  - `src`의 `n`개의 바이트를 dst에 복사함
  - 리턴 값은 `dst`
  - `dst`와 `src`는 겹치면 안 됨
- `void *memmove(void *dst, const void *src, size_t n)`
  - `memcpy`와 같으나 `dst`와 `src`는 겹칠 수 있음
- `int memcmp(const void *cs, const void *ct, int n)`
  - `cs`와 `ct`의 처음 `n`개의 바이트를 비교함
  - 리턴 값은 `strcmp`와 같음
- `void *memset(void *dst, int c, int n)`
  - `c`를 `dst`의 처음 `n`개의 바이트에 넣음
  - 리턴 값은 `dst`

<stdlib.h>: 유틸리티 함수

- `double atof(const char *s)`
  `int atoi(const char *s)`
  `long atol(const char *s)`
  - s를 **double**, **int**, **long**로 변환함
- `int rand(void)`
  - 0~RAND_MAX(32767 이상)의 랜덤 수를 리턴함
- `int srand(unsigned int seed)`
  - seed를 난수 발생기의 seed로 설정함

<stdlib.h>: 종료시키는 함수

- `void abort(void)`
  - 프로그램을 그 상태에서 정지시켜 버림
- `void exit(int status)`
  - 프로그램을 정상적인 상태로 종료시킴
  - `status`는 `0(EXIT_SUCCESS)`인 경우 프로그램이 정상적으로 끝나게 된다는 의미
  - `status`는 `0(EXIT_FAILURE)`이 아닌 경우 프로그램이 오류로 인해 끝나게 되다는 의미

<stdlib.h>: 종료시키는 함수

- `int atexit(void (*fcn)(void))`
  - 프로그램 종료 시 함수 `fcn`을 호출할 수 있는 상태로 해줌
  - 에러 시 `0`이 아닌 값을 리턴함
- `int system(const char *s)`
  - 문자열 `s`에 저장되는 명령을 실행함
  - `s`는 `NULL`이 아닌 경우 `s`에 저장되는 명령을 실행하고 그 명령이 리턴하는 값을 리턴함
  - `s`는 `NULL`이 인 경우 `0`이 아닌 값을 리턴함

<stdlib.h>: 탐색 및 정렬

- `void bsearch(const void *key, const void *base, size_t n, size_t size, int (*cmp)(const void *keyval, const void *datum))`
  - `*key`와 맞는 대상을 `base[0]~base[n-1]`에서 찾음
  - `cmp` 함수는 배열의 요소를 비교하는 것을 담당함
  - 배열 `base`는 오름차순으로 정렬되어 야 함
  - `*key`와 일치하는 항목이 리턴되고 일치하는 것이 없으면 `NULL`이 리턴됨
- `void qsort(void *base, size_t n, size_t size, int (*cmp)(const void*, const void*))`
  - 배열 `base`를 오름차순으로 정렬함
  - `cmp` 함수는 배열의 요소를 비교하는 것을 담당함

<assert.h>: 진단

- `void assert(int expression)`
  - `expression`이 `0`이면 다음과 같은 문장이 `stderr`에 출력됨

    Assertion failed: `expression`, file `filename`, line `nnn`
  - `abort`에 의해 프로그램이 종료되고, 파일 이름과 행번호는 `__FILE__`과 `__LINE__`의 두 매크로에 의해 주어짐
  - `expression`이 `0`이 아니면 아무 초지를 취하지 않음
  - 프로그램 디버그 시 많이 쓰임

<stdarg.h>: 매개변수 리스트

- 형이나 숫자를 알 수 없는 매개변수를 처리하는 동작을 하도록 해줌
- 여러 매개변수가 있는 함수 `f`의 이름 붙은 마지막 매개변수가 `lastarg`이라고 가정함
  - `va_list ap`
    - `f`의 내부에서 변수 `ap`를 `va_list` 형으로 선언하면 `ap`는 매개변수를 차례로 가리키게 됨
  - `va_start(va_list ap, lastarg)`
    - `ap`를 초기화해줌
    - `ap`를 사용하기 전에 `va_start` 매크로를 사용해야 함
  - `type va_arg(va_list ap, type)`
    - 이름 없는 매개변수와 같은 형 같은 값의 숫자(또는 문자)가 리턴함
    - `ap`는 같이 움직여서 다음번 `va_arg`를 실행할 때 다음 매개변수를 처리하도록 됨
  - `va_end(va_list ap)`
    - 함수 `f`가 끝나기 전에 실행해야 함

<stdarg.h>: 매개변수 리스트

- 예,

```
int sum(int num, …) {
    va_list ap;
    int total=0;
    va_start(ap, num);
    while (num>0) {
        total += va_arg(ap, int);
        num--;
    }
    va_end(ap);
    return total;
}

int a = sum(4,1,2,3,4); // a=10
int b = sum(2,1,2); // b=3
```

<time.h>: 날짜 및 시간

- clock_t, time_t는 프로그램 실행시간을 알려주고, struct tm은 달력 표시의 시간을 알려줌

`int tm_sec;`	초(0~59)
`int tm_min;`	분(0~59)
`int tm_hour;`	시간(0~23)
`int tm_mday;`	날짜(1~31)
`int tm_mon;`	달(0~11)
`int tm_year;`	연도(1900부터)
`int tm_wday;`	요일(0~6, 0은 일요일)
`int tm_yday;`	날짜(0~365)
`int tm_isdst;`	일광 절약시간 사용 여부 양수이면 사용을 뜻하고, 0이면 아님을 뜻하고, 음수이면 정보가 틀린 것을 뜻함

<time.h>: 날짜 및 시간

- `clock_t clock(void)`
  - 프로그램 실행 시작부터의 시간을 리턴함
  - 에러 시 `-1`을 리턴함
  - 시간을 초로 바꾸려면 `clock()/CLK_TCK`를 계산하면 됨
- `time_t time(time_t *tp)`
  - 현재 시간을 알려줌
  - 시간이 잘못되어 있으면 `-1`을 리턴함
  - `tp`가 `NULL`이 아니면 리턴 값은 `*tp`에 할당됨
- `double difftime(time_t t1, time_t t2)`
  - `t2-t1`을 초로 계산함

<time.h>: 날짜 및 시간

- `time_t mktime(struct tm *tp)`
  - `*tp`에 있는 `struct tm`(지역시간) 대상을 `time_t`(표준시간)로 변환해줌
  - 변환이 불가능하면 `-1`을 리턴함
- `char *asctime(const struct tm *tp)`
  - `*tp`에 있는 시간을 다음 형태의 문자열로 바꿈

```
Sun Jan 3 15: 14: 13 1988\n\0
```

- `struct tm *localtime(const time_t *tp)`
  - `time_t`(표준시간)를 `struct tm`(지역시간)로 변환해줌
- `char *ctime(const time_t *tp)`
  - 표준시간을 지역시간을 나타내는 문자열로 변환해줌
  - `asctime(localtime(tp))`와 같음

<time.h>: 날짜 및 시간

- `size_t strftime(char *s, size_t smax, const char *fmt, const struct tm *tp)`
  - `*tp`의 형태로 되어 있는 정보를 `fmt`의 형태인 `s`로 변환해줌
  - `s`의 최대 문자 수는 `smax`에 의해 정해지고, `smax`보다 많은 수의 문자가 만들어지면 리턴 값은 `0`이 됨

%a	요일 이름 약자
%A	요일 이름
%b	달 이름 약자
%B	달 이름
%d	날짜(01~31)
%H	시간(00~23)
%I	시간(01~12)
%m	달(01~12)
%M	분(00~59)
%p	오전, 오후 표시
%S	초(00~59)