## 8.2. 헤더와 라이브러리

### 해더(Header)

- 헤더 파일은 확장자가 `.h` 인 파일로, 함수 선언, 매크로 정의, 상수, 전역 변수 등을 포함합니다.
- 프로그램에서 헤더 파일을 포함시키면, 그 파일 안에 정의된 내용을 사용할 수 있습니다. 이는 `#include` 지시어를 통해 이루어집니다.
- 예를 들어, `#include <stdio.h>`는 표준 입력 및 출력 함수를 사용하기 위해 필요한 표준 라이브러리 헤더입니다.

### 라이브러리

- 라이브러리는 재사용 가능한 함수나 데이터의 모음입니다. 이들은 이미 컴파일된 형태(바이너리 코드)로 존재하며, 프로그램에서 직접 호출하여 사용할 수 있습니다.
- C 언어의 표준 라이브러리는 다양한 기본 함수들(예: printf, scanf)을 제공하여 프로그래머가 복잡한 작업을 쉽게 처리할 수 있도록 돕습니다.

OS와 관련한 라이브러리 문제

#### 파일 경로 처리

- 윈도우 시스템에서는 파일 경로 구분자로 백슬래시(`\`)를 사용하는 반면, 유닉스/리눅스 시스템에서는 슬래시(`/`)를 사용합니다.
- 예를 들어, C 언어의 파일 처리 함수들(`fopen`, `fread`, `fwrite` 등)에서 경로를 지정할 때 이러한 차이로 인해 문제가 발생할 수 있습니다.

#### 줄 바꿈 문자

- 윈도우에서는 줄 바꿈을 나타내기 위해 캐리지 리턴(`\r`)과 라인 피드(`\n`) 두 문자를 사용합니다(`\r\n`).
- 반면, 유닉스/리눅스에서는 라인 피드(`\n`)만을 사용합니다.
- 이 차이로 인해 텍스트 파일을 다른 OS로 옮겼을 때 줄 바꿈이 제대로 표시되지 않는 문제가 발생할 수 있습니다.

ANSI C에서 strdup 함수는 마치 표준처럼 사용하지만 제공되지 않을 경우도 있다.

```
# strdup 함수 구현
#include <stdlib.h>
#include <string.h>

char *strdup(const char *s) {
    char *d = malloc(strlen(s) + 1);   // 메모리 할당 (문자열 길이 + 1)
    if (d != NULL) {
        strcpy(d, s);                  // 문자열 복사
    }
```

```
    return d;
}
```

```
#ifdef _OLD_C
        extern int fread();
        extern int fwrite();

#else
# if edfined(__STDC__) || defiend(__cpluplus)
        extern size_t fread(void*, size_t, size_t, FILE*);
        extern size_t fwrite(const void*, size_t, size_t, FILE*);
# else /* not __STDC__ || __cplusplus */
        extern size_t fread();
        extern size_t fwrite();
# endif /* else not __STDC__ || __cplusplus */
#endif
```

구조화된 헤더 파일은 난해하고 유지보수가 어렵다

- 컴파일러와 환경마다 다른 헤더 파일을 사용하면 별도로 여러 파일을 관리 해야 하지만 각 파일이 자급 자족형이 되고 특정 시스템에 적합한 정의를 제공하며 엄격한 ANSI C 표준 환경에서 strdup 함수를 include 하는 것과 같은 실수의 발생 확률을 줄인다

standard_io.h

```
#include <stdio.h>
extern size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);
extern size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);
```

nonstandard_io.h

```
extern int fread();
extern int fwrite();
```

old_c_io.h

```
extern int fread();
extern int fwrite();
```

헤더 파일은 프로그램에 있는 함수와 같은 이름의 함수를 선언해서 이름 공간을 오염시킬 수 있다.

```
/* 몇몇 stdio 헤더가 wprintf 이름을 사용하기 때문에 재정의하여 없엔다*/
#define wprintf stdio_wprintf
```

```
#include <stdio.h>
#undef wprintf
/*우리가 만든 wprintf()를 사용하는 코드 ...*/
```

이 코드처럼 헤더 파일에 나오는 모든 wprintf 부분을 stdio_wprintf로 바꿔서 우리가 만든 wprintf와 헷갈리지 않게 한다
싸우는 것보다 바꾸는 편이 쉬울 뿐 아니라, 분명히 더 안전하다.

프로그래밍 언어의 정의에 속하지 않는 표준들은 더 광범위하게 사용되고 더 잘 수립된 표준을 선택하고, 가장 중심적이고 흔히 쓰이는 측면들을 지키면 호환성이 높아진다.

최근 라이브러리 때문에 고생한 예

```
안녕하세요 *** 님
어제 말씀해주신 오류를 바탕으로 디버깅 한 결과에 대해 전달 드리겠습니다.

기존 코드 에러 로그 확인 결과 GPU사용 한정이 아닌 CPU사용 시에도 에러 발생을 확인하여, 아래와 같은 순서로 디버깅 진행하였습니다

문제 : 파이썬코드 실행 중 torch 호출시 오류 발생
 -> 우분투 20.04 default Python Version인 3.8 버전 다운그레이
1. 파이썬 버전  3.8 로 다운그레 후 코드실행 (미니콘다 설치 후 가상환경사용)
-> CPU모드로 코드에 진입은 가능하나 에러 발생 (3번~)
2. wsl2에서는 우분투를 제외하고 nvidia-driver 사용이 불가능
-> GPU 모드 미지원

CPU 모드 진입 시 문제 해결
3. tesseract 설치는 했으나, 언어팩 없는 오류 발생
-> git hub 사이트에서 언어팩 설치
4. 언어팩을 설치 했으나, Leptonica 버전이 달라 함수가 바뀌거나 사라져 dependency를 맞추기 어려움
-> tesseract에 맞는 Leptonica 버전 확인 및 centos 9 stream 지원여부 확인

위와 같은 제약사항으로 우분투 20.04 OS사용을 권장드립니다.
추가적으로 궁금하신 사항 문의주시면 답변 드리겠습니다.
```