3장 C 프로그래밍 환경

리눅스 시스템 프로그래밍

청주대학교 전자공학과
한철수

목차

- 컴파일러 사용법
- make 시스템
- 디버거
- 이클립스 통합개발환경

유닉스/리눅스와 C 언어

- 유닉스/리눅스 시스템은 C 언어와 밀접하게 연관되어 있음.
  - 유닉스/리눅스 자체가 C 언어로 작성되어 있음.
  - 많은 유틸리티와 상용 프로그램들이 C 언어로 작성되어 있음.
- 따라서 유닉스/리눅스 시스템을 깊이 있게 이해하기 위해서는 반드시 C 언어를 알아야 함.

단일 모듈 프로그램

- 단일 모듈 프로그램
  - 하나의 소스 파일로 이루어진 C 프로그램
  - 간단한 프로그램인 경우에 하나의 모듈(파일)로 작성할 수 있음.
- 실습
  - P.123 프로그램 3.8 (longest.c)
    - 키보드 입력을 받아 입력된 문장 중에서 가장 긴 문장을 longest 배열에 복사하고, ^D를 입력하면 longest 배열에 저장된 문장을 출력하고 종료하는 프로그램.

longest.c

```
#include <stdio.h>
#include <string.h>
#define MAXLINE 100

char line[MAXLINE];
char longest[MAXLINE];
void copy(char from[], char to[]);

int main()
{
    int len=0;
    int max=0;

    while (fgets(line, MAXLINE, stdin) != NULL) {
        len = strlen(line);

        if (len > max) {
            max = len;
            copy(line, longest);
        }
    }
    if (max > 0)
        printf("%s", longest);

    return 0;
}

void copy(char from[], char to[])
{
    int i=0;

    while ((to[i] = from[i]) != '₩0')
        ++i;
}
```

컴파일러 사용법

- gcc (GNU C compiler)
  - 오픈 소스 C 컴파일러로서 널리 사용되고 있음.
- 가장 간단한 gcc 사용법

```
$ gcc longest.c                 // 실행 파일 a.out이 만들어짐.
$ ./a.out                       // 실행.
```

- -c 옵션

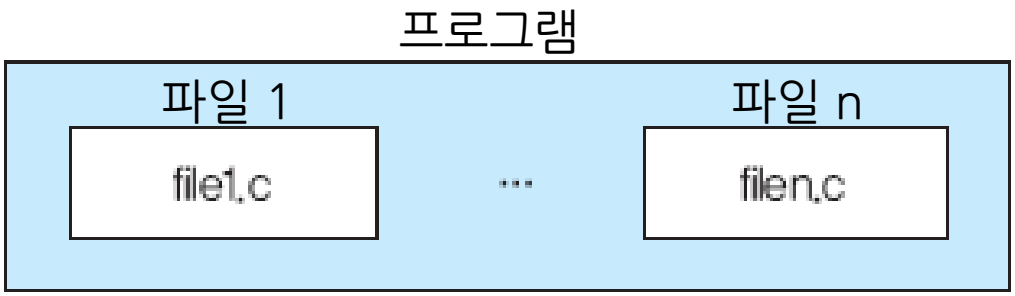
```
                                // 컴파일만 수행함.
$ gcc -c longest.c              // 목적파일 longest.o를 만듦.
```

- -o 옵션

```
                                // 출력 파일의 이름을 지정함.
$ gcc -o longest longest.o      // 목적파일로 실행 파일을 만듦.
      혹은
$ gcc -o longest longest.c      // 소스코드로 실행 파일을 만듦.
$ ./longest                     // 실행.
```

다중 모듈 프로그램

- 단일 모듈 프로그램의 문제점
  - 코드의 재사용이 어려움.
  - 여러 사람이 참여하여 프로그래밍하기 어려움.
- 다중 모듈 프로그램
  - 여러 개의 .c 파일들로 이루어진 프로그램
  - 일반적으로 큰 프로그램인 경우에 다중 모듈 프로그램으로 작성함.



다중 모듈 프로그램의 작성 예

- copy 함수를 공유하기 위해 main 프로그램과 copy 함수를 따로 분리하여 별도 파일로 작성함.
  - main.c // main 프로그램
  - copy.c // copy 함수의 소스 코드
  - copy.h // copy 함수의 헤더 파일
    보통 헤더파일에는 함수 프로토타입 선언,
    매크로 상수에 대한 정의 등을 작성함.

main.c

```
#include <stdio.h>
#include <string.h>
#include "copy.h"

char line[MAXLINE];
char longest[MAXLINE];

int main()
{
    int len=0;
    int max=0;
    while (fgets(line, MAXLINE, stdin) != NULL) {
        len = strlen(line);

        if (len > max) {
            max = len;
            copy(line, longest);
        }
    }

    if (max > 0)
        printf("%s", longest);

    return 0;
}
```

copy.h

```
#define MAXLINE 100
void copy(char from[], char to[]);
```

copy.c

```
#include <stdio.h>
#include "copy.h"

void copy(char from[], char to[])
{
    int i=0;

    while ((to[i] = from[i]) != '₩0')
        ++i;
}
```

다중 모듈 프로그램의 컴파일 방법

- 방법 1
  - 파일을 각각 컴파일 한 후 실행 파일을 생성하는 방법

```
$ gcc -c main.c
$ gcc -c copy.c
$ gcc -o main main.o copy.o
```

- 방법 2
  - 한번에 실행 파일을 생성하는 방법

```
$ gcc -o main main.c copy.c
```

- 실행

```
$ ./main
```

make 시스템

- 대규모 프로그램의 경우에는 헤더 파일, 소스 파일, 목적 파일, 실행 파일의 모든 관계를 기억하고 체계적으로 관리하는 것이 필요함.
- make 시스템을 이용하면 이러한 작업을 효과적으로 할 수 있음.

Makefile

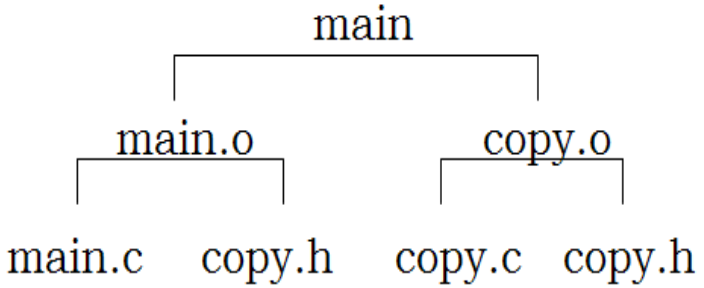
- 실행 파일을 만들기 위해서 사용되는 파일들 사이의 상호 의존 관계 및 실행 파일을 만드는 방법 등을 기술한 텍스트 파일
- 파일 이름에는 .make(혹은 .mk)라는 확장자를 붙이거나, 확장자 없이 Makefile 또는 makefile을 파일 이름으로 사용함.
- 메이크파일의 일반적인 구성 형식

  대상: 의존리스트

  <tab> 명령리스트

- 예: Makefile

```
main: main.o copy.o
	gcc -o main main.o copy.o
main.o: main.c copy.h
	gcc -c main.c
copy.o: copy.c copy.h
	gcc -c copy.c
```



make 시스템의 실행

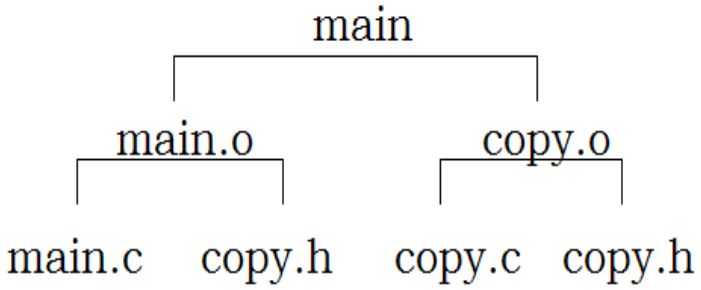
- $ make [-f 메이크파일]
  - f 옵션이 없으면 현재 디렉터리내의 Makefile 혹은 makefile을 실행함.
- make 실행 예

  $ make

  gcc -c main.c

  gcc -c copy.c

  gcc -o main main.o copy.o



- copy.c 파일을 수정한 후, 재실행 예

  $ make

  gcc -c copy.c

  gcc -o main main.o copy.o

  - make 시스템을 이용하면 변경된 파일과 관련된 파일들만 컴파일하고 실행파일을 만들기 때문에, 컴파일 시간을 크게 단축시킬 수 있음.

디버거

- 프로그램을 디버깅하기 위한 소프트웨어
- 대표적인 디버거
  - Gdb (GNU Debugger)

이클립스 통합개발환경

- 이클립스(Eclipse)는 다양한 플랫폼에서 동작하는 무료 통합개발환경(IDE)으로서 여러 컴퓨터 언어들(C/C++, Java 등)을 지원함.

질문

Q&A