

## 컴파일 환경[3]

### □ Makefile과 make

- 소스 파일이 여러 개를 묶어서 실행파일을 생성하는 도구
- make 명령은 Makefile의 내용에 따라 컴파일, /usr/ccs/bin을 경로에 추가해야함

```
# vi ~/.profile
.....
PATH=$PATH:/usr/local/bin:/usr/ccs/bin
export PATH
```

[예제 1-3]

ex1\_3\_main.c

```
01 #include <stdio.h>
02 extern int addnum(int a, int b);
03
04 int main(void) {
05     int sum;
06
07     sum = addnum(1, 5);
08     printf("Sum 1~5 = %d\n", sum);
09
10     return 0;
11 }
```

[예제 1-3]

ex1\_3\_addnum.c

```
01 int addnum(int a, int b) {
02     int sum = 0;
03
04     for (; a <= b; a++)
05         sum += a;
06     return sum;
07 }
```



# 컴파일 환경[3]

## [예제 1-3] make 명령 사용하기

## Makefile

```
01 # Makefile
02
03 CC=gcc
04 CFLAGS=
05 OBJS=ex1_3_main.o ex1_3_addnum.o
06 LIBS=
07 all: add
08
09 add: $(OBJS)
10     $(CC) $(CFLAGS) -o add $(OBJS) $(LIBS)
11
12 ex1_3_main.o: ex1_3_main.c
13     $(CC) $(CFLAGS) -c ex1_3_main.c
14 ex1_3_addnum.o: ex1_3_addnum.c
15     $(CC) $(CFLAGS) -c ex1_3_addnum.c
16
17 clean:
18     rm -f $(OBJS) add core
```

ex1\_3\_main.c와  
ex1\_3\_addnum.c를  
묶어서 add라는  
실행파일 생성

```
# make
gcc -c ex1_3_main.c
gcc -c ex1_3_addnum.c
gcc -o add ex1_3_main.o
ex1_3_addnum.o

# ls
Makefile  add*  ex1_3_addnum.c
ex1_3_addnum.o  ex1_3_main.c
ex1_3_main.o

# add
Sum 1~5 = 15
```

# 오류 처리 함수[1]

## □ 오류 메시지 출력 : perror(3)

```
#include <stdio.h>
void perror(const char *s);
```

### [예제 1-4] perror 함수 사용하기

ex1\_4.c

```
01 #include <sys/errno.h>
02 #include <unistd.h>
03 #include <stdlib.h>
04 #include <stdio.h>
05
06 int main(void) {
07     if (access("unix.txt", R_OK) == -1) {
08         perror("unix.txt");
09         exit(1);
10     }
11
12     return 0;
13 }
```

```
# ex1_4.out
unix.txt: No such file or directory
```

## 오류 처리 함수[2]

### □ 오류 메시지 출력 : strerror(3)

```
#include <string.h>
char *strerror(int errnum);
```

#### [예제 1-5] strerror 함수 사용하기

ex1\_5.c

```
01 #include <sys/errno.h>
02 #include <unistd.h>
03 #include <stdlib.h>
04 #include <stdio.h>
05 #include <string.h>
06
07 extern int errno;
08
09 int main(void) {
10     char *err;
11
12     if (access("unix.txt", R_OK) == 1) {
13         err = strerror(errno);
14         printf("오류:%s(unix.txt)\n", err);
15         exit(1);
16     }
17
18     return 0;
19 }
```

오류에 따라  
메시지를 리턴

```
# ex1_5.out
오류: No such file or directory(unix.txt)
```

# 동적 메모리 할당[1]

## □ 메모리할당 : malloc(3)

```
#include <stdlib.h>
void *malloc(size_t size);
```

- 인자로 지정한 크기의 메모리 할당

```
char *ptr
ptr = malloc(sizeof(char) * 100);
```

## □ 메모리할당과 초기화 : calloc(3)

```
#include <stdlib.h>
void *calloc(size_t nelem, size_t elsize);
```

- **nelem \* elsize** 만큼의 메모리를 할당하고, 0으로 초기화

```
char *ptr
ptr = calloc(10, 20);
```



## 동적 메모리 할당[2]

### □ 메모리 추가 할당: realloc(3)

```
#include <stdlib.h>
void *realloc(void *ptr, size_t size);
```

- 이미 할당받은 메모리(ptr)에 size 크기의 메모리를 추가로 할당

```
char *ptr, *new;
ptr = malloc(sizeof(char) * 100);
new = realloc(ptr, 100);
```

### □ 메모리 해제 : free(3)

```
#include <stdlib.h>
void free(void *ptr);
```

- 사용을 마친 메모리 반납



# 명령행 인자[1]

## □ 명령행 : 사용자가 명령을 입력하는 행

- 명령행 인자 : 명령을 입력할 때 함께 지정한 인자(옵션, 옵션인자, 명령인자 등)
- 명령행 인자의 전달 : main 함수로 자동 전달

```
int main(int argc, char *argv[])
```

### [예제 1-6] 명령행 인자 출력하기

ex1\_6.c

```
01  #include <stdio.h>
02
03  int main(int argc, char *argv[]) {
04      int n;
05
06      printf("argc = %d\n", argc);
07      for (n = 0; n < argc; n++)
08          printf("argv[%d] = %s\n", n, argv[n]);
09
10      return 0;
11  }
```

```
# ex1_6.out -h 100
argc = 3
argv[0] = ex1_6.out
argv[1] = -h
argv[2] = 100
```

