기본 라이브러리 함수

POSIX 표준 라이브러리 함수

- **POSIX (Portable Operating System Interface) 표준**
  - **IEEE 에서 1988년에 UNIX 라이브러리를 위한 POSIX.1 발표**
  - **1998년 The Open Group, IEEE POSIX, ISO/IEC 연합기술운영회의 회원들로 구성된 Austin 그룹 결성**
  - **2001, 2002년 IEEE 표준 1003.1-2001 제정**
  - 기타 표준 구성
    - **POSIX.2 -** 표준 쉘과 유틸리티 인터페이스
    - **POSIX.4 -** 쓰레드 관리
    - **POSIX.5 - Ada** 언어를 위한 바인딩
    - **POSIX.6 -** 보안 기능

표준 C 라이브러리 함수들

C 언어 표준 라이브러리

헤더 파일	함수 기능	표준 함수
stdio.h	표준입출력	printf(); scanf(); putchar(); getchar(); ...
	화일입출력	fopen(); fclose(); fprintf(); ...
stdlib.h	수치변환	atoi(); itoa(); ...
	난수발생	rand(); srand(); ...
	탐색 및 정렬	bsearch(); qsort();...
ctype.h	문자변환	tolower(); toupper(); ...
	문자판별	isalpha(); isdigit(); isupper(); ...
String.h	문자열처리	strcpy(); strlen(); strcmp(); ...
	동적메모리관리	memcpy(); memset(); memchr(); ...
Math.h	수학함수	sin(); cos(); sqrt(); ...
Search.h	탐색	lsearch(); hsearch(); tsearch(); ...
Time.h	시간관련함수	time(); difftime(); ctime(); ...

표준 C 라이브러리 함수

▣ 표준 C 라이브러리 함수 사용 예

◈ 표준 C 라이브러리 함수 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: library.c
 */
#include <stdio.h>
#include <math.h>
#include <ctype.h>

int main(void)
{
    int i=4;
    char c='A';

    printf("sqrt(%d) = %f₩n", i, sqrt(i));
    printf("tolower(%c) = %c₩n", c, tolower(c));
}
```

◈ 프로그램 실행 결과

```
[cprog2@seps5 ch7]$ gcc -o library library.c -lm
[cprog2@seps5 ch7]$ ./library
sqrt(4) = 2.000000
tolower(A) = a
[cprog2@seps5 ch7]$
```

표준 C 라이브러리 함수

▣ 표준 C 라이브러리 함수 사용 예

◈ 표준 C 라이브러리 함수 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: testsqrt.c
 */

1:    #include <stdio.h>
2:                        /* sqrt()함수를 사용하기 위한 헤더파일 <math.h> 미포함 */
3:    int main(void)
4:    {
5:        int i=4;
6:        printf("sqrt(%d) = %f₩n", i, sqrt(i));
7:    }
```

◈ 프로그램 실행 결과

```
[cprog2@seps5 ch7]$ gcc testsqrt.c
testsqrt.c: In function `main':
testsqrt.c:9: warning: return type of `main' is not `int'
/tmp/ccYDwg2S.o(.text+0x25): In function `main':
: undefined reference to `sqrt'
collect2: ld returned 1 exit status
```

프로그램 인터페이스

main() 함수의 매개변수

Main() 함수의 기본형

```
main(int argc, char *argv[])
{
   프로그램 명령들
}
```

사용 예

```
$ main a b c
```

main ⟶ argv[0]
a ⟶ argv[1]
b ⟶ argv[2]
c ⟶ argv[3]

Main() 함수의 매개변수

▣ 매개변수 사용 예

◈ 매개변수 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: main.c
 */

#include <stdio.h>
int main(int argc, char *argv[])
{
        int i;
        for (i=0; i<argc; i++) printf("%s ", argv[i]);
}
```

◈ 매개변수 예제 프로그램 실행 결과

```
[cprog2@seps1 cprog2]$ gcc -o main main.c
[cprog2@seps1 cprog2]$ ./main main string 10
main string 10
[cprog2@seps1 cprog2]$
```

환경변수 처리

▣ 환경변수 처리

◈ main() 함수의 확장형

```
main(int argc, char *argv[], char *envp[])
{
   프로그램 명령들
}
```

◈ 많이 사용하는 환경변수

환경변수	설명
USER	로그인한 사용자 이름
LOGNAME	프로세스와 관련된 로그인 사용자 이름
HOME	사용자의 로그인 디렉토리
LANG	LC_ALL 등이 지정되지 않았을 때의 로케일 이름
LC_ALL	우선적으로 지정되는 로케일 이름. LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_MONETARY, LC_NUMERIC, LC_TIME 등을 포함한다.
PATH	실행 파일을 찾는 디렉토리들의 목록
PWD	현재 작업 디렉토리
SHELL	사용자의 로긴 쉘 파일 이름
TERM	출력 터미널 타입
TMPDIR	임시 디렉토리 경로
LD_LIBRARY_PATH	동적 로딩/링킹을 위한 라이브러리 경로

환경변수 처리

▣ 환경변수 처리 사용 예

◈ 환경변수 처리 예제 프로그램

```
/*
  * 7장 기본 라이브러리 함수
 * 파일이름: envp.c
 */
#include <stdio.h>
int main(int argc, char *argv[], char *envp[])
{
   int i;
   for (i=0; envp[i]; i++)
         printf("%s₩n", envp[i]);
}
```

◈ 환경변수 처리 예제 프로그램 실행 결과

```
[cprog2@seps1 cprog2]$ gcc -o envp envp.c
[cprog2@seps1 cprog2]$ ./envp
HOSTNAME=....
TERM=xterm
....
_=./envp
[cprog2@seps1 cprog2]$
```

환경변수 처리

환경변수 처리의 또다른 방법

environ 전역변수 이용

```
#include <stdlib.h>

extern char **environ;
```

환경변수 처리 함수들

기능

- **getenv()** 함수는 환경변수에 대한 값을 반환
- **putenv()** 함수는 환경변수를 추가하거나 값을 변경
- **setenv()** 함수는 환경변수를 추가하거나 값을 변경
- **unsetenv()** 함수는 환경변수를 제거

사용법

```
#include <stdlib.h>
char *getenv(const char *name);
int putenv(char *string);
int setenv(const char *name, const char *value, int overwrite);
void unsetenv(const char *name);
```

- **getenv()** 는 성공 시 환경변수 값에 대한 주소를 반환, 실패 시 **NULL** 값을 반환
- **putenv()** 와 **setenv()** 는 성공 시 **0** 을 반환하고 실패시 **-1**을 반환

프로그램 옵션 처리

▣ 프로그램 옵션 처리 함수

◈ 기능

- **getopt()** 함수는 명령 라인 옵션을 파싱

◈ 사용법

```
#include <unistd.h>
int getopt(int argc, char * const argv[], const char *optstring);
extern char *optarg;
extern int optind, opterr, optopt;
```

- 성공 시 옵션 문자를 반환하고, 옵션 중 하나가 빠진 문자가 있으면 ‘:’, 모르는 옵션 문자가 있으면 ‘?’, 옵션 목록의 마지막이면 **-1** 을 반환
- 옵션 : “-” 로 시작하는 인자
- **getopt()** 함수가 반복해서 호출될 때마다, 옵션 문자 각각을 계속 반환
- 처리할 다음 명령 라인 인자의 인덱스는 외부 변수 **optind** 에 갱신

프로그램 옵션 처리

▣ 프로그램 옵션 처리 함수

◈ Optstring : 프로그램에서 처리 가능한 옵션들을 명시하는 문자열

- 추가 인자가 필요한 옵션인 경우 ‘:’을 붙이고, 추가 인자가 옵션일 경우 ‘::’ 을 붙임
- 예) **optstring** 가 **“abc:d::”**인 경우, 옵션으로 **‘-a’, ‘-b’, ‘-c추가인자’, ‘-d[추가인자]’** 가 가능
- 추가 인자에 대한 주소는 외부 변수 **optarg** 에 저장
- **optstring**을 ‘+’로 시작하거나 **POSIXLY_CORRECT** 환경변수가 설정되어 있으면, 옵션이 아닌 인자를 발견하자마자 처리를 중단
- **optstring**이 ‘-’ 로 시작하면 옵션이 아닌 인자들은 문자 코드 **1**을 가진 옵션 인자처럼 처리
- ‘--’ 는 옵션의 끝을 강제로 설정하는 특별한 인자

◈ optstring에 명시되지 않은 옵션을 만난 경우

- 에러메시지를 표준 오류로 출력하고 외부 변수 **optopt** 에 옵션 문자를 저장하고 **'?'**을 반환
- 외부변수 **opterr** 이 미리 **0** 으로 설정하면 오류 메시지를 출력 않음

프로그램 옵션 처리

▣ 프로그램 옵션 처리 사용 예

◈ 프로그램 옵션 처리 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: testopt.c
 */
#include <unistd.h>
#include <stdio.h>
#define VERSION "0.0.1"
void print_help();

int main(int argc, char *argv[])
{
    int index, c;

    opterr = 0;
    if (argc == 1)
        print_help();
    while ((c = getopt (argc, argv, "hvf:")) != -1)
        switch (c) {
        case 'h':
            print_help();
            exit(0);
        case 'v':
            printf("testopt %s₩n", VERSION);
            exit(0);
        case 'f':
            printf("File: %s₩n", optarg);
            break;
        case '?':
            fprintf (stderr, "Unknown option '-%c'.₩n", optopt);
            print_help();
            exit(1);
        }
    for (index = optind; index < argc; index++)
        printf ("Non-option argument %s₩n", argv[index]);
}
void print_help()
{
    printf("testopt (ver.%s) is a getopt example.₩n", VERSION);
    printf("testopt [-h] [-v] [-f FILE]₩n₩n");
    printf("  -h            print this help and exit₩n");
    printf("  -v            print version and exit₩n");
    printf("  -f FILE       set file₩n");
}
```

프로그램 옵션 처리

▣ 프로그램 옵션 처리 사용 예

◈ 프로그램 옵션 처리 예제 프로그램 실행 예

```
[cprog2@seps1 cprog2]$ gcc -o testopt testopt.c
[cprog2@seps1 cprog2]$ ./testopt
testopt (ver.0.0.1) is a getopt example.
testopt [-h] [-v] [-f FILE]

  -h              print this help and exit
  -v              print version and exit
  -f FILE          set file
[cprog2@seps1 cprog2]$ ./testopt -v
testopt 0.0.1
[cprog2@seps1 cprog2]$ ./testopt -f file
File: file
[cprog2@seps1 cprog2]$ ./testopt -a help
Unknown option '-a'.
testopt (ver.0.0.1) is a getopt example.
testopt [-h] [-v] [-f FILE]

  -h              print this help and exit
  -v              print version and exit
  -f FILE          set file
[cprog2@seps1 cprog2]$ ./testopt help
Non-option argument help
[cprog2@seps1 cprog2]$
```

수치 연산 함수

▣ 수치 연산 함수

◈ UNIX/LINUX 제공 수치 연산 함수들은 수학 라이브러리 (libm) 에 포함

함수	기능	인수 형	함수 형
abs(k)	k의 절대치	int	int
acos(x)	x의 아크코사인, x는 -1 ~ 1의 범위	double	double
asin(x)	x의 아크사인, x는 -1 ~ 1의 범위	int	int
atan(x)	x의 아크탄젠트	double	double
ceil(x)	x보다 큰 최소 정수		
cos(x)	x의 코사인, x의 단위는 radian		
cosh(x)	x의 쌍곡선.코사인		
exp(x)	지수 ex		
fabs(x)	x의 절대치		
floor(x)	x보다 크지 않은 최대정수		
loge(x)	자연대수 logex		
log10(x)	상용대수 log10x		
pow(x,y)	누승xy		
sin(x)	x의 사인, x의 단위는 radian		
sinh(x)	x의 쌍곡선.사인		
sqrt(x)	평방근 √x		
tan(x)	x의 탄젠트, x의 단위는 radian		
tanh(x)	x의 쌍곡선.탄젠트		

수치 연산 함수

▣ 수치 연산 함수 사용 예

◈ 수치 연산 함수 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: testmath.c
 */

#include <stdio.h>
#include <math.h>

int main(void) /*산술함수*/
{
    double x;
    printf("%10s%10s%10s%15s%20s₩n",
            "x","log(x)","sqrt(x)","exp(x)","pow(10,0,x)");
    for (x=1;x<=10;x++)
      printf("%10.5f10.5f%10.5f%15.5f%20.5₩n",
            x,log(x),sqrt(x),exp(x),pow(10.0,x));
}
```

x	log(x)	sqrt(x)	exp(x)	pow(10.0,x)
1.00000	0.00000	1.00000	2.71828	10.00000
2.00000	0.69315	1.41421	7.38906	100.00000
3.00000	1.09861	1.73205	20.08554	1000.00000
4.00000	1.38629	2.00000	54.59815	10000.00000
5.00000	1.30944	2.23607	148.41316	100000.00000
6.00000	1.78176	2.44949	403.42879	1000000.00000
7.00000	1.94591	2.64575	1096.63316	10000000.00000
8.00000	2.07944	2.82843	2980.95799	100000000.00000
9.00000	2.19722	3.00000	8103.08393	1000000000.00000
10.00000	2.30259	3.16228	22026.46579	10000000000.00000

◈ 프로그램 실행 결과

난수 관련 함수

▣ 난수 관련 함수

◈ 기능

- **rand() 함수는 0에서 RAND_MAX 사이의 난수를 생성.**
- **srand() 함수는 새로운 seed 수로부터 난수열을 생성하도록 설정**

◈ 사용법

```
#include <stdlib.h>
int rand(void);
void srand(unsigned int seed);
```

- **rand() 함수는 0에서 RAND_MAX 사이의 정수 난수를 하나 반환**
  - RAND_MAX 는 32 비트 정수인 경우 2147483647 이지만, 32767 만큼 낮게 지정 가능
  - 0 ~ 1.0 실수 난수 생성 : (double) rand()/(RAND_MAX+1.0)
- **srand(seed)는 seed 수를 난수열의 개시점으로 설정**
  - 이후에 호출되는 rand() 가 반환하는 값은 새로운 난수열을 시작

난수 관련 함수

난수 관련 함수 사용 예

난수 관련 함수 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: testrand.c
 */
#include <stdio.h>
#include <stdlib.h>

int main(void) /* 주사위 */
{
    int i,x;
    for (i=0;i<20;i++){
        x=(int)(6*(rand()/(RAND_MAX+1.0))+1);
        printf("%d ",x);
    }
}
```

프로그램 실행 결과

```
[cprog2@seps1 cprog2]$ gcc -o testrand testrand.c
[cprog2@seps1 cprog2]$ ./testrand
6 3 5 5 6 2 3 5 2 4 3 4 3 4 6 6 4 5 1 4
[cprog2@seps1 cprog2]$
```

문자 검사/변환 함수

▣ 문자 검사 함수 (is~)

◈ 기능

- ➢ 특정 문자에 대하여 검사를 수행하여 그 결과를 반환

◈ 사용법

```
#include <ctype.h>
int is~(int c);
```

함수	설명
isalpha(c)	문자 c가 영문자('A'~'Z', 'a'~'z')이면 참
isalnum(c)	문자 c가 영문/수자('A'~'Z', 'a'~'z', '0'~'9')이면 참
isascii(c)	문자 c가 ASCII 코드 (0~0x7f)이면 참
isblank(c)	문자 c가 blank 문자 (스페이스 또는 탭)이면 참
iscntrl(c)	문자 c가 콘트롤 코드(0~0x1f, 0x7f)이면 참
isdigit(c)	문자 c가 숫자 문자('0'~'9')이면 참
isgraph(c)	문자 c가 0x21 ~ 0x7e이면 참
islower(c)	문자 c가 영문소문자('a'~'z')이면 참
isprint(c)	문자 c가 인쇄문자 가능문자 (0x20~0x7e)이면 참
ispunct(c)	문자 c가 구독점 (0x21~0x2f, 0x3a ~ 0x40, 0x5b ~ 0x60, 0x7b~0x7e)이면 참
isspace(c)	문자 c가 스페이스 (0x09~0x0d, 0x20)이면 참
isupper(c)	문자 c가 영문대문자('A'~'Z')이면 참
isxdigit(c)	문자 c가 16진 문자('0'~'9', 'A'~'F', 'a'~'f')이면 참

문자 검사/변환 함수

▣ 문자 변환 함수 (to~)

◈ 기능

- ➢ 특정 문자를 다른 계층의 문자로 변환
- ➢ **tolower()** 함수는 특정 문자가 대문자이면 소문자로 변환
- ➢ **toupper()** 함수는 특정 문자가 소문자이면 대문자로 변환
- ➢ **toascii()** 함수는 특정 문자를 7 비트 아스키 문자로 변환

◈ 사용법

```
#include <ctype.h>
int tolower(int c);
int toupper(int c);
int toascii(int c);
```

문자 검사/변환 함수

▣ 문자 검사/변환 함수 사용 예

◈ 문자 변환 함수 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: tofunc.c
 */

#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main (void) /*소문자·대문자 변환*/
{
    int i;
    char s[20], t[20];

    printf("string ? ");
    scanf("%s", s);
    for (i=0;i<=strlen(s);i++){
        if (isupper(s[i]))
            t[i]=tolower(s[i]);
        else if (islower(s[i]))
            t[i]=toupper(s[i]);
        else
            t[i]=s[i];
    }
    printf("%s ----> %s₩n", s, t);
}
```

◈ 프로그램 실행 결과

```
[[cprog2@seps1 cprog2]$ gcc -o tofunc tofunc.c
[cprog2@seps1 cprog2]$ ./tofunc
string  ?  Hello
Hello ----> hELLO
[cprog2@seps1 cprog2]$
```

문자열 처리 함수

- 문자열 처리 함수
  - 기능
    - 다양한 문자열 처리
  - 사용법

```
#include <string.h>

size_t strlen(const char *s);
char * strcpy(char *dest, const char *src);
char * strcat(char *dest, const char *src);
int strcmp(const char *s1, const char *s2);
int strcasecmp(const char *s1, const char *s2);
char * strchr(const char *s, int c);
char * strstr(const char *s1, const char *s2);
char *strpbrk(const char *s1, const char *s2);
size_t strspn(const char *s1, const char *s2);
size_t strcspn(const char *s1, const char *s2);
char *strtok(char *s1, const char *s2);
```

함수	설명
strlen()	문자열 s 의 길이를 반환
strcpy()	배열 s 에 문자열 t 를 복사
strcat()	배열 s 에 문자열 t 를 연결하여 붙임
strcmp()	문자열 s1 과 s2 를 비교
strcasecmp()	문자열 s1과 s2를 대소문자에 관계없이 비교
strchr()	문자열 s에서 문자 c가 처음 나타나는 위치를 반환
strstr()	문자열 s1에서 문자열 s2가 처음 나타나는 위치를 반환
strpbrk()	문자열 s1에서 s2의 한 문자가 처음 나타나는 위치 반환
strspn()	문자열 s1에서 s2에 속하는 문자만을 포함하는 부분의 길이 반환
strcspn()	문자열 s1에서 s2의 어떤 문자도 포함하지 않는 부분의 길이반환
strtok()	문자열 s1을 s2의 구분자를 이용하여 토큰으로 분리

문자열 처리 함수

▣ 문자열 처리 함수 사용 예

◈ 문자열 처리 함수 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: testmin.c
 */

#include <stdio.h>
#include <string.h>

int main(void) /*문자열 처리*/
{
    char s[20],t[20],min[20];
    printf("char 1 ? "); scanf("%s",s);
    printf("char 2 ? "); scanf("%s",s);

    if (strcmp(s,t)<0)
        strcpy(min,s);
    else
        strcpy(min,t);

    printf("%s₩n", min);
}
```

◈ 프로그램 실행 결과

```
[cprog2@seps1 cprog2]$ gcc -o testmin testmin.c
[cprog2@seps1 cprog2]$ ./testmin
char 1 ? candy
char 2 ? ann
ann
[cprog2@seps1 cprog2]$
```

탐색 및 정렬 함수

▣ 배열 탐색 함수

◈ 기능

- **lfind()** 함수는 배열에 대해 선형 탐색을 수행
- **lsearch()** 함수는 배열에 대해 선형 탐색을 수행하고, 찾는 항목이 없으면 배열의 끝에 새로운 항목으로 추가

◈ 사용법

```
#include <search.h>
void *lfind(const void *key, const void *base, size_t *nmemb, size_t size,
comparison_fn_t compar);
void *lsearch(const void *key, void *base, size_t *nmemb, size_t size,
comparison_fn_t compar);
```

- **key** 가 나타내는 객체인 첫째 인자와 배열 요소인 둘째 인자 간의 비교를 수행하여 같으면 **0**, 첫째 인자가 둘째 인자보다 크면 양수, 작으면 음수 값을 반환
- 일치하는 항목이 없는 경우 **lfind()** 함수는 **NULL** 을 반환하는 반면, **lsearch()** 함수는 **key** 객체를 배열의 끝에 추가하고 **nmemb** 를 하나 증가시키고 추가한 항목의 주소를 반환

배열 탐색 함수

▣ 배열 탐색 함수

◈ compar : 비교 함수

- 2개의 **const void *** 형 인자를 가지고 정수값을 반환하는 함수 정의
- 예) **double** 값을 비교하는 함수

```
int comparison_fn_t (const void *, const void *);

int compare_doubles (const void *a, const void *b) {
      const double *da = (const double *) a;
      const double *db = (const double *) b;

      return (*da > *db) - (*da < *db);
}
```

배열 탐색 함수

▣ 배열 탐색 함수 사용 예

◈ 배열 탐색 함수 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: testlsearch.c
 */
#include <stdio.h>
#include <search.h>
#define TABLESIZE 5

int  compare(const void *ap, const void *bp)
{
    return ( *(int *)ap - *(int *)bp);
}
int main()
{
   int table[TABLESIZE] = {1,2,3,4,5};
   unsigned int n=TABLESIZE;
   int item, *ptr;

   item = 6;
   ptr = (int *)lsearch(&item, table, &n, sizeof(int), compare);
   if (ptr >= table + n - 1)
         printf("%d is not in the table(1-%d), but added.₩n",  item, n);
   else
         printf("%d is in the table(1-%d).₩n", *ptr, n);

   item = 7;
   ptr = (int *) lfind (&item, table, &n, sizeof(int), compare);
   if (ptr == NULL)
         printf("%d is not in the table(1-%d).₩n", item, n);
   else
         printf("%d is in the table(1-%d).₩n", *ptr, n);
}
```

◈ 프로그램 실행 결과

```
[cprog2@seps1 cprog2]$ gcc -o testlsearch testlsearch.c
[cprog2@seps1 cprog2]$ ./testlsearch
6 is not in the table(1-6), but added.
7 is not in the table(1-6).
[cprog2@seps1 cprog2]$
```

정렬된 배열 탐색 함수

▣ 정렬된 배열 탐색 함수

◈ 기능

- **bsearch()** 함수는 정렬된 배열에 대하여 이진 탐색을 수행

◈ 사용법

```
#include <stdlib.h>
void *bsearch(const void *key, const void *base, size_t nmemb, size_t
size, comparison_fn_t compar);
```

- 각각 **size** 크기의 **nmemb** 개의 항목을 가진 정렬된 배열 **base** 에 대하여 **key** 가 가리키는 내용에 대하여 이진 탐색 **(binary search)** 을 수행

정렬된 배열 탐색 함수

▣ 정렬된 배열 탐색 함수 사용 예

◈ 정렬된 배열 탐색 함수 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: testbsearch.c
 */
#include <stdio.h>
#include <search.h>
#define TABLESIZE 5

int  compare(const void *ap, const void *bp)
{
    return ( *(int *)ap - *(int *)bp);
}
```

```
int main()
{
    int table[SIZE] = {1,2,3,4,5};
    int n=TABLESIZE, item=6, *ptr;

    ptr = (int *)bsearch(&item, table, n, sizeof(int), compare);
    if (ptr == NULL)
        printf("%d is not in the table(1-%d).₩n", item, n);
    else
        printf("%d is in the table(1-%d).₩n", *ptr, n);
}
```

◈ 프로그램 실행 결과

```
[cprog2@seps1 cprog2]$ gcc -o testbsearch testbsearch.c
[cprog2@seps1 cprog2]$ ./testbsearch
6 is not in the table(1-5).
[cprog2@seps1 cprog2]$
```

배열 정렬 함수

▣ 배열 정렬 함수

◈ 기능

- **qsort()** 함수는 배열에 대하여 정렬

◈ 사용법

```
#include <stdlib.h>

void qsort (void *base, size_t nmemb, size_t size, comparison_fn_t compar);
```

- **size** 크기의 **nmemb** 개의 항목을 가진 배열 **base** 에 대하여 비교 함수를 통하여 오름차순으로 빠른 정렬 **(quick sort)** 을 수행
- 만약 비교하는 두 항목이 같으면 정렬 순서는 알 수 없음
- **qsort()** 함수는 반환되는 값을 가지지 않음

배열 정렬 함수

▣ 배열 정렬 함수 사용 예

◈ 배열 정렬 함수 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: testqsort.c
 */
#include <stdio.h>
#include <search.h>
#define TABLESIZE 5

int  compare(const void *ap, const void *bp)
{
    return ( *(int *)ap - *(int *)bp);
}
```

```
int main()
{
    int table[TABLESIZE] = {4,2,1,5,3};
    int i;

    qsort (table, TABLESIZE, sizeof(int), compare);
    for (i=0; i < TABLESIZE; i++)
        printf("%d ", table[i]);
    printf("₩n");
}
```

◈ 프로그램 실행 결과

```
[cprog2@seps1 cprog2]$ gcc -o testqsort testqsort.c
[cprog2@seps1 cprog2]$ ./testqsort
1 2 3 4 5
[cprog2@seps1 cprog2]$
```

해쉬 탐색 함수

- 배열 탐색과 해쉬 탐색
  - 선형/이진 탐색 - 배열 크기가 커짐에 따라 탐색 시간이 많이 걸림
  - 해쉬 탐색 - 배열 크기가 고정된 경우, 해쉬 테이블과 키를 이용하여 빠른 탐색 가능
- 해쉬 탐색 함수
  - 기능
    - **hcreate()** 함수는 해쉬 테이블을 생성
    - **hdestroy()** 함수는 해쉬 테이블을 삭제
    - **hsearch()** 함수는 해쉬 테이블로부터 특정한 항목을 탐색
  - 사용법

```
#include <search.h>
int hcreate(size_t nel);
void hdestroy(void);
ENTRY *hsearch(ENTRY item, ACTION action);
```

    - **hcreate()** 함수는 해쉬 테이블에 대한 메모리 할당이 실패하면 **0**, 그렇지 않으면 **0** 이 아닌 값을 반환
    - **hdestroy()** 함수는 반환값 없음
    - **hsearch()** 함수는 탐색한 항목을 반환하거나, **action** 이 **ENTER** 이고 해쉬 테이블이 꽉 차 있거나 **action** 이 **FIND** 이고 항목을 찾을 수 없으면 **NULL** 을 반환

해쉬 탐색 함수

▣ 해쉬 탐색 함수

◈ 해쉬 탐색을 위한 구조체

- ➢ 해쉬 테이블의 각 항목과 탐색을 위한 키 **(key)** 로 구성

```
typedef struct entry {
        char *key;      /* 탐색을 위한 키 문자열 */
        void *data;     /* 데이터 */
} ENTRY;
```

◈ **hsearch()** 함수

- ➢ **item** 과 같은 키 값을 가진 항목을 해쉬 테이블로부터 탐색
- ➢ 탐색이 성공하면 찾은 항목의 키와 값을 가진 구조체의 주소를 반환
- ➢ 탐색이 실패한 경우
  - – 해쉬 테이블이 꽉 차 있으면 NULL 을 반환
  - – action 이 ENTER 이면 새로운 항목이 item 과 같은 키 값을 가지며 해쉬 테이블에 삽입되고 새로 추가된 항목에 대한 주소가 반환
  - – action 이 FIND 이면 NULL 을 반환

해쉬 탐색 함수

▣ 해쉬 탐색 함수 사용 예

◈ 해쉬 탐색 함수 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: testhsearch.c
 */
#include <stdio.h>
#include <search.h>

struct info {
        int id, age;
};
#define TABLESIZE  50

int main( )
{
    char nametable[TABLESIZE*20];
    char *nameptr = nametable; /* 이름테이블에서 다음 이
름 */
    struct info infotable[TABLESIZE];
    struct info *infoptr = infotable; /* info테이블에서 다음 in
fo */
    ENTRY item, *found;
    char name[30];
    int i = 0;
    /* 해쉬 테이블 생성 */
    (void) hcreate(TABLESIZE);
    while (scanf("%s%d%d", nameptr, &infoptr->id, &infoptr-
>age) != EOF  && i++ < TABLESIZE) {
        item.key = nameptr;
        item.data = (char *)infoptr;

        /* 해쉬 테이블에 넣기 */
        (void) hsearch(item, ENTER);

        nameptr += strlen(nameptr) + 1;
        infoptr++;
    }

    /* 해쉬 테이블 탐색 */
    item.key = name;
    while (scanf("%s", item.key) != EOF) {
        if ((found = hsearch(item, FIND)) != NULL) {
            printf("found %s, id=%d, age=%d₩n",
                found->key,
                ((struct info *)found->data)->id,
                ((struct info *)found->data)->age);
        } else {
            printf("no such employee %s₩n", name);
        }
    }
}
```

해쉬 탐색 함수

▣ 해쉬 탐색 함수 사용 예

◈ 해쉬 탐색 함수 예제 프로그램 실행 결과

```
[cprog2@seps1 cprog2]$ gcc -o testhsearch testhsearch.c
[cprog2@seps1 cprog2]$ ./testhsearch
HongGilDong 1 30
JangGilSan 2 40
ImGgukJung 3 45
ImGgukJung <---ctrl-D 입력
found ImGgukJung, id=3, age=45
HwangJinEe
no such employee HwangJinEe <---ctrl-D 입력
[cprog2@seps1 cprog2]$
```

트리 탐색 함수

- 트리 탐색
  - 해쉬 탐색처럼 효율적인 탐색 수행; 유닉스/리눅스는 이진 트리 **(binary tree)** 함수 지원
- 트리 탐색 함수
  - 기능
    - **tsearch()** 함수는 이진 트리로부터 특정한 항목을 탐색
    - **tfind()** 함수는 이진 트리로부터 특정한 항목을 탐색
    - **tdelete()** 함수는 이진 트리로부터 특정한 항목을 삭제
    - **twalk()** 함수는 이진 트리를 방문
    - **tdestroy()** 함수는 이진 트리를 삭제
  - 사용법

```
#include <search.h>
void * tsearch(const void *key, void **rootp, comparison_fn_t compar);
void * tfind(const void *key, void *const *rootp, comparison_fn_t compar);
void * tdelete (const void *key, void **rootp, comparison_fn_t compar);
void twalk (const void *root, __action_fn_t action);
void tdestroy (void *vroot, void *(free_node)(void *nodep));
```

    - **tsearch()** 함수는 탐색한 항목, 새로 추가한 항목, 또는 **NULL**을 반환
    - **tfind()** 함수는 탐색한 항목 또는 **NULL**을 반환
    - **tdelete()** 함수는 삭제된 항목의 부모, 또는 **NULL** 을 반환
    - **twalk()** 와 **tdestroy()** 함수는 반환값 없음

트리 탐색 함수

트리 탐색 함수

tsearch() 와 tfind() 함수

- **item** 과 같은 키 값을 가진 항목을 트리로부터 탐색
- 탐색하는 항목이 트리에 존재하면 그 항목에 대한 주소를 반환
- 일치하는 항목이 없는 경우
  - tfind() 함수는 NULL 을 반환
  - tsearch() 함수는 key 객체를 트리에 추가하고 그 항목의 주소를 반환

tdelete() 함수

- 트리로부터 **key** 객체를 탐색하여 삭제
- 탐색하는 항목이 트리에 없는 경우에는 **NULL** 을 반환

twalk() 함수

- 트리의 **root** 노드를 시작으로 깊이 우선 **(depth-first)** 왼쪽에서 오른쪽으로 각 노드를 방문하며 특정한 동작을 수행
- 둘째 인자 **action** 함수

```
typedef enum {preorder, postorder, endorder, leaf} VISIT;
void __action_fn_t (const void *nodep, VISIT value, int level);
```

  - nodep : 방문하는 노드의 주소
  - value : 현재 노드의 상태; “자식이 없는 노드 (leaf)” 인 경우 한 번 방문, “내부 노드 (internal node)” 인 경우 첫째 자식을 방문하기 전 (preorder), 첫째 자식을 방문하고 나서 둘째 자식을 방문하기 전 (postorder), 둘째 자식을 방문한 후 (endorder) 모두 세 번 방문
  - level : 트리의 깊이 수준

트리 탐색 함수

▣ 트리 탐색 함수 사용 예

◈ 트리 탐색 함수 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: testtsearch.c
 */
#include <search.h>
#include <stdio.h>
#include <string.h>
struct node {
    char *name;
    int age;
};
#define TABLESIZE 50

char nametable[TABLESIZE*20];   /* 이름을 저장할 테이블 */
struct node nodetable[TABLESIZE]; /* 저장할 노드 테이블 */
struct node *root = NULL;    /* 루트 노드 */

int compare(const void *cp1, const void *cp2)
{
   return strcmp(((struct node *)cp1)->name, ((struct node *
)cp2)->name);
}
void print_node(const void *, VISIT, int);
```

```
int main()
{
    char *nameptr = nametable;
    struct node *nodeptr = nodetable;
    struct node **ret;
    int i = 0;

    while (scanf("%s%d", nameptr, &nodeptr->age) != EOF
&& i++ < TABLESIZE) {
        nodeptr->name = nameptr;

        /* 트리에 넣기 */
        ret = (struct node **) tsearch((void *) nodeptr,
                                (void **) &root, compare);
        printf("₩"%s₩" 님이 ", (*ret)->name);
        if (*ret == nodeptr)
            printf("트리에 추가되었습니다.₩n");
        else
            printf("트리에 이미 존재합니다.₩n");
        nameptr += strlen(nameptr) + 1;
        nodeptr++;
    }
    twalk((void *) root, print_node);
}
```

트리 탐색 함수

▣ 트리 탐색 함수 사용 예

◈ 트리 탐색 함수 예제 프로그램 계속

```
/* twalk 가 노드를 처음 만날때 출력 */
void print_node(const void *nodeptr, VISIT order, int level)
{
    if (order == preorder || order == leaf)
        printf("이름 = %-20s, 나이 = %d₩n",
            (*(struct node **)nodeptr)->name,
            (*(struct node **)nodeptr)->age);
}
```

◈ 프로그램 실행 결과

```
[cprog2@seps1 cprog2]$ gcc -o testtsearch testtsearch.c
[cprog2@seps1 cprog2]$ ./testtsearch
[cprog2@seps1 cprog2]$ ./testtsearch
HongGilDong 32
"HongGilDong" 님이 트리에 추가되었습니다.
JangGilSan 40
"JangGilSan" 님이 트리에 추가되었습니다.
ImGgukJung 45
"ImGgukJung" 님이 트리에 추가되었습니다. <--- Ctrl-D 입력
이름 = ImGgukJung          , 나이 = 45
이름 = HongGilDong         , 나이 = 32
이름 = JangGilSan          , 나이 = 40
[cprog2@seps1 cprog2]$
```

동적 메모리 관리

▣ 메모리 할당

◈ 정적 할당(static allocation)

➢ 변수 선언 시 미리 필요한 만큼의 메모리를 할당

```
char name[10];
```

◈ 동적 할당(dynamic allocation)

➢ 프로그램 실행 도중에 변수에 필요한 만큼의 메모리를 할당

동적 메모리 할당과 반환

▣ 동적 메모리 할당과 반환 함수

◈ 기능

- **malloc()** 함수는 **num** 바이트 수 만큼 메모리 공간을 할당
- **calloc()** 함수는 크기**(size)*** 개수**(num)** 만큼 메모리 공간을 할당한 다음 메모리 공간을 **0**이나 **\0**로 채움
- **free()** 함수는 할당된 메모리 공간을 반환

◈ 사용법

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

- **malloc()** 및 **calloc()** 함수는 성공 시 할당된 메모리의 포인터 반환 , 실패 시 **NULL**값을 반환
- **free()** 함수는 반환값 없음

동적 메모리 할당과 반환

▣ 동적 메모리 할당과 반환 함수

◈ 필요한 만큼의 메모리를 동적으로 할당하여 메모리의 낭비를 줄임

```
#include <stdio.h>
#include <stdlib.h>
main()
{
  int len;
    ........
  scanf("%d", &len);
    ........
  /* 필요한 크기 만큼 동적으로 메모리 할당 */
  name = (char *)malloc(len * sizeof(char));
    ........
}
```

◈ **malloc()** 은 **free()** 와 쌍으로 사용

```
........
 scanf("%d", &len);
   ........
 name = (char *)malloc(len * sizeof(char));
   ........
 free(name);
   ........
```

동적 메모리 할당과 반환

▣ 동적 메모리 할당과 반환 함수 사용 예

◈ 동적 메모리 할당과 반환 함수 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: testmalloc.c
 */
#include <stdio.h>
#include <stdlib.h>
main()
{
    int max, i;
    int *ptr;

    printf("배열의 원소 개수는? ");
    scanf("%d",&max);

    /* 사용자가 입력한 수만큼 메모리 할당 */
    ptr = (int *)malloc(max * sizeof(int));

    /* 메모리 할당에 실패할 경우 */
    if (ptr == NULL){
      printf("메모리를 할당하지 못했습니다.");
      exit(-1);
    }

    for( i = 0; i < max; i++ )
      scanf("%d", &ptr[i]);

    printf("입력 숫자 리스트:");
    for( i = 0; i < max; i++ )
      printf("%d ", *(ptr+i));
    printf("₩n");
    free((int *)ptr);
}
```

동적 메모리 할당과 반환

▣ 동적 메모리 할당과 반환 함수 사용 예

◈ 프로그램 실행 결과

```
[cprog2@seps1 cprog2]$ gcc -o testmalloc testmalloc.c
[cprog2@seps1 cprog2]$ ./testmalloc
배열의 원소 개수는? 3
12
23
43
입력 숫자 리스트:12 23 43
[cprog2@seps1 cprog2]$
```

◈ 배열의 각 원소에 접근하기 위해서는 포인터 형식을 사용 또는 배열의 첨자 인덱스 형식 모두를 사용 가능

```
ptr[0] = *ptr
ptr[1] = *(ptr+1)
        :
        :
ptr[9] = *(ptr+9)
```

동적 메모리 재할당 함수

▣ 동적 메모리 재할당 함수

◈ 기능

- **malloc() 함수나 calloc()함수에 의해 이미 할당된 메모리 기억 공간에 대해 블록의 크기를 변경**

◈ 사용법

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

- **ptr : 변경할 메모리 블록의 주소**
  - 충분한 기억 공간이 존재한다면 realloc() 함수는 추가로 메모리를 할당하고 포인터 ptr을 반환
  - 충분한 메모리 내의 기억 공간이 존재하지 않으면, size로 지정된 바이트 크기만큼의 새로운 블록이 할당되고, 이전 메모리 블록의 데이터는 새로운 블록으로 복사되고, 이전 메모리 블록은 해체되고 realloc() 함수는 새로운 메모리 블록에 대한 포인터를 반환
  - ptr이 NULL 이면 함수는 malloc() 처럼 사용
- **size : 할당할 메모리 크기**
  - size 값이 0이라면 포인터 ptr이 지적하는 메모리가 해체되고, NULL을 반환
- 시스템 메모리 전 영역에서 메모리 재할당을 수행하기 위한 충분한 메모리가 존재하지 않으면 **NULL**을 반환하고 원래의 메모리 블록을 변경하지 않는다.

동적 메모리 재할당 함수

▣ 동적 메모리 재할당 함수 사용 예

◈ 동적 메모리 재할당 함수 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: testrealloc.c
 */
#include <stdio.h>
#include <stdlib.h>
main()
{
    char c, *ptr;
    int count;

    ptr=NULL;

    count = 0;
    while((c=getchar()) != '₩n'){
        if (count == 0 )
            ptr = (char *)malloc(1);
        else
            ptr = (char *)realloc(ptr, count+1); /* 메모리재할당 */
        *(ptr+count)=c;
        count++;
    }
    *(ptr+count)='₩0';

    printf("입력문자열 : %s 총 할당메모리:%d₩n",ptr, count);
    free(ptr);
}
```

◈ 프로그램 실행 결과

```
[cprog2@seps1 cprog2]$ gcc -o testrealloc testrealloc.c
[cprog2@seps1 cprog2]$ ./testrealloc
I am a boy.
입력문자열 : I am a boy. 총 할당메모리:11
[cprog2@seps1 cprog2]$
```

메모리 처리 함수

▣ 메모리 처리 함수

◈ 기능

- **memset()** 함수는 메모리 영역 **s** 를 **n** 크기만큼 **c** 값으로 설정
- **memcpy()** 함수는 메모리 영역 **src** 를 **n** 크기만큼 **dest** 에 복사
- **memmove()** 함수는 메모리 영역 **src** 를 **n** 크기만큼 **dest** 에 복사
- **memcmp()** 함수는 메모리 영역 **s1** 과 **s2** 를 비교
- **memchr()** 함수는 메모리 영역 **s**에서 문자 **c**가 처음 나타나는 위치 반환

◈ 사용법

```
#include <string.h>
void * memset(void *s, int c, size_t n);
void * memcpy (void * dest, const void * src, size_t n);
void * memmove (void * dest, const void * src, size_t n);
int memcmp(const char *s1, const char *s2, size_t n);
void * memchr(const char *s, int c, size_t n);
```

메모리 처리 함수

▣ 메모리 처리 함수 사용 예

◈ 메모리 처리 함수 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: testmem.c
 */
#include <stdio.h>
#include <string.h>

int main ()
{
        char srcstr[] = "This is a test string.";
        char dststr[25], *ptr;
        int ret;

        memcpy (dststr, srcstr, strlen(srcstr)+1);
        printf("dststr is ₩"%s₩"₩n", dststr);

        memset (srcstr + 5, 'a', 5);
        printf("srcstr is ₩"%s₩"₩n", srcstr);

        ret = memcmp (srcstr, dststr, strlen(srcstr));
        if (ret > 0)       printf("srcstr is greater than dststr.₩n");
        else if (ret == 0) printf("srcstr is equal to dststr.₩n");
        else               printf("srcstr is less than dststr.₩n");

        ptr = memchr (dststr, ' ', strlen(dststr));
        if (ptr != NULL)
                printf ("First space was at position %d.₩n", ptr-dststr+1);
        else
                printf ("Space was not found.₩n");
}
```

◈ 프로그램 실행 결과

```
[cprog2@seps1 cprog2]$ gcc -o testmem testmem.c
[cprog2@seps1 cprog2]$ ./testmem
dststr is "This is a test string."
srcstr is "This aaaaatest string."
srcstr is less than dststr.
First space was at position 5.
[cprog2@seps1 cprog2]$
```

시간 관련 함수

시간 표현 함수

기능

- **time()** 함수는 **EPOCH** 시 이후부터 현재까지의 초 단위로 측정한 시간을 반환
- **EPOCH - 1970년 1월 1일 00:00** 시 **UTC(Coordinated Universal Time)** 로 정의

사용법

```
#include <time.h>
time_t time(time_t *t);
```

- 성공 시 초 단위의 시간을 반환, 실패 시 **-1** 값을 반환
- 달력 시간을 나타내는 **time_t** 형은 보통 **long** 으로 정의

시간 비교 함수

기능

- **difftime()** 함수는 두 개의 시간 값을 비교하여 그 차이를 반환

사용법

```
#include <time.h>
double difftime(time_t time1, time_t time2);
```

시간 표현 함수

▣ 시간 표현 함수 사용 예

◈ 현재 시간 출력 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: curtime.c
 */
#include <stdio.h>
#include <time.h>
main()
{
      timt_t curtime;

      time(&curtime);
      printf("current time is %d₩n", curtime);
}
```

◈ 프로그램 실행 결과

```
[cprog2@seps1 cprog2]$ gcc -o curtime curtime.c
[cprog2@seps1 cprog2]$ ./curtime
current time is 1048201394
[cprog2@seps1 cprog2]$
```

시간 변환 함수

▣ 시간 변환 함수

◈ 기능

- **ctime()** 은 **time_t** 형 시간 정보를 문자열로 변환
- **gmtime()** 과 **localtime()**은 **time_t** 형 시간 정보를 **struct tm** 형으로 변환
- **asctime()** 은 **struct tm** 형 시간 정보를 문자열로 변환
- **mktime()** 은 **struct tm** 형 시간 정보를 **time_t** 형으로 변환

◈ 사용법

```
#include <time.h>
char *ctime(const time_t *timep);
struct tm *gmtime(const time_t *timep);
struct tm *localtime(const time_t *timep);
char *asctime(const struct tm *tm);
time_t mktime(struct tm *tm);
```

- **ctime()** 함수의 문자열
  "요일 월 날짜 **00:00:00** 년도" 형식
- **struct tm** 형은
  일, 달, 년, 요일 등의 요소로 구성

```
struct tm {
    int tm_sec;  /* 초 seconds (0~59) */
    int tm_min;  /* 분 minutes (0~59) */
    int tm_hour; /* 시 hours (0~23) */
    int tm_mday;/* 일 day of the moon (1~31) */
    int tm_mon;  /* 월 month (0~11) */
    int tm_year; /* 년 year (1900년부터 시작) */
    int tm_wday;/* 요일 day of the week (0~6) */
    int tm_yday; /* 1월 1일부터의 날짜 day of the year */
    int tm_isdst; /* 일광 절약 시간 daylight saving time */
}
```

시간 변환 함수

▣ 시간 변환 함수 사용 예

◈ 현재 시간 변환 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: testtime.c
 */
#include <stdio.h>
#include <time.h>
main()
{
        time_t curtime;
        struct tm *tp, *gtp;
        time(&curtime);
        tp = localtime(&curtime);
        printf("ctime: %s",ctime(&curtime));
        printf("localtime: %d:%d:%d₩n",tp->tm_hour,tp->tm_min,
tp->tm_sec);
        gtp = gmtime(&curtime);
        printf("gmtime: %d:%d:%d₩n", gtp->tm_hour, gtp->tm_min,
gtp->tm_sec);
        printf("asctime: %s",asctime(tp));
        printf("mktime: time is %d₩n",mktime(tp));
}
```

◈ 프로그램 실행 결과

```
[cprog2@seps1 cprog2]$ gcc -o testtime testtime.c
[cprog2@seps1 cprog2]$ ./testtime
ctime: Tue Jan 10 16:58:55 2006
localtime: 16:58:55
gmtime: 7:58:55
asctime: Tue Jan 10 07:58:55 2006
mktime: time is 1136847535
[cprog2@seps1 cprog2]$
```

세밀한 시간 측정 함수

▣ 세밀한 시간 측정 함수

◈ 기능

- **gettimeofday()** 함수는 마이크로초 단위의 시간과 시간대**(timezone)**을 반환

◈ 사용법

```
#include <sys/time.h>
int gettimeofday(struct timeval *tv, struct timezone *tz);
```

- 성공 시 마이크로초 단위의 시간과 시간대를 반환, 실패 시 **-1** 값 반환
- **timeval** 구조체

```
struct timeval {
        time_t    tv_sec;    /* 초 */
        time_t    tv_usec;   /* 마이크로초 */
}
```

- **timezone** 구조체

```
struct timezone {
        int       tz_minuteswest;     /* 그리니치 서측 분차 */
        int       tz_dsttime;         /* DST 보정시간 */
}
```

세밀한 시간 측정 함수

▣ 세밀한 시간 측정 함수 사용 예

◈ 세밀한 현재 시간 측정 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: curtime.c
 */
#include <stdio.h>
#include <sys/time.h>

main()
{
        struct timeval tv;
        struct timezone tz;
        int ret;

        ret=gettimeofday(&tv, &tz);
        if (ret == 0) {
                printf("time = %u.%06u, minuteswest = %d, dsttime
= %d₩n",
                        tv.tv_sec, tv.tv_usec,
                        tz.tz_minuteswest, tz.tz_dsttime);
        } else {
                perror("gettimeofday() failed.₩n");
                exit(1);
        }
}
```

◈ 프로그램 실행 결과

```
[cprog2@seps1 cprog2]$ gcc -o gettimeofday gettimeofday.c
[cprog2@seps1 cprog2]$ ./gettimeofday
time = 1137743102.516952, minuteswest = -540, dsttime = 0
[cprog2@seps1 cprog2]$
```

프로세스 시간 측정 함수

▣ 프로세스 시간 측정 함수

◈ 기능

- **times()** 함수는 프로세스의 수행 시간을 얻는다

◈ 사용법

```
#include <sys/times.h>
clock_t times(struct tms *buf);
```

- 성공 시 시스템 부팅 후 **clock tick** 을 반환
- **tms** 구조체
  - 모든 단위는 clock tick

```
struct tms {
        clock_t tms_utime;  /* 사용자의 CPU 수행 시간 */
        clock_t tms_stime;  /* 시스템의 CPU 수행 시간 */
        clock_t tms_cutime; /* 자식 프로세스에서 사용자의 CPU 수행 시간 */
        clock_t tms_cstime; /* 자식 프로세스에서 시스템의 CPU 수행 시간 */
};
```

프로세스 시간 측정 함수

▣ 프로세스 시간 측정 함수 사용 예

◈ 프로세스 시간 측정 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: processtime.c
 */
#include <stdio.h>
#include <unistd.h>
#include <sys/times.h>

int main()
{
      double cticks;
      clock_t tcstart, tcend;
      struct tms tmstart, tmend;
      int i, a = 1, b = 2, c;

      if ((tcstart = times(&tmstart)) == -1) {
            perror("Failed to get start time");
            exit(1);
      }
      printf("Fraction of CPU time used is %d₩n", tcstart);
      printf("CPU time spent executing process is %d₩n",  tmstart.tms_utime);
      printf("CPU time spent in the system is %d₩n",  tmstart.tms_stime);
      for (i=0; i<10000000; i++)
            c = a + b;
      if ((tcend = times(&tmend)) == -1) {
            perror("Failed to get start time");
            exit(1);
      }
      printf("Fraction of CPU time used is %d₩n", tcend);
      printf("CPU time spent executing process is %d₩n",  tmend.tms_utime);
       printf("CPU time spent in the system is %d₩n",tmend.tms_stime);
      cticks = tmend.tms_utime + tmend.tms_stime - tmstart.tms_utime - tmst
art.tms_stime;
      printf("Total CPU time is %f seconds.₩n", cticks/100.);
      printf("Fraction of CPU time used is %f₩n",  cticks/(tcend - tcstart));
}
```

프로세스 시간 측정 함수

▣ 프로세스 시간 측정 함수 사용 예

◈ 프로그램 실행 결과

```
[cprog2@seps1 cprog2]$ gcc -o processtime processtime.c
[cprog2@seps1 cprog2]$ ./processtime
Fraction of CPU time used is 497024066
CPU time spent executing process is 0
CPU time spent in the system is 0
Fraction of CPU time used is 497024069
CPU time spent executing process is 2
CPU time spent in the system is 0
Total CPU time is 0.020000 seconds.
Fraction of CPU time used is 0.666667
[cprog2@seps1 cprog2]$
```

sleep 함수

sleep 함수

기능

- **sleep()** 함수는 특정한 시간(초) 가 경과하거나, 시그널을 받을 때까지 현재 프로세스수행을 중지

사용법

```
#include <unistd.h>
unsigned int sleep(unsigned int seconds);
```

- 경과한 시간이 지나 **0** 을 반환하거나, 인터럽트에 의해 끝난 경우 남은 시간을 반환
- 초 단위 이상으로 보다 세밀한 시간 동안 프로세스 실행을 중지하기 위해서 **sleep()** 함수 외에 **usleep()** 함수나 **nanosleep()** 함수를 제공

sleep 함수

▣ sleep 함수 사용 예

◈ sleep 예제 프로그램

```
/*
 * 7장 기본 라이브러리 함수
 * 파일이름: testsleep.c
 */
#include <stdio.h>
#include <time.h>

main()
{
        time_t tstart, tend;
        double diff;

        time(&tstart);
        sleep (3);
        time(&tend);
        diff = difftime(tend, tstart);
        printf("sleep time is %lf.₩n", diff);
}
```

◈ 프로그램 실행 결과

```
[cprog2@seps1 cprog2]$ gcc -o testsleep testsleep.c
[cprog2@seps1 cprog2]$ ./testsleep
sleep time is 3.000000.
[cprog2@seps1 cprog2]$
```