함수	시스템 포함 파일	함수 프로토타입	설명		
assert	assert.h	void assert(int expression);	진단 메세지를 인쇄하고, 표현식이 false인 경우 프로그램을 종료합니다.		
isalnum	ctype.h	int isalnum(int c);	*c가 영숫자인지 테스트합니다.*		
isalpha	ctype.h	int isalpha(int c);	*c가 영문자인지 테스트합니다.*		
isascii4	ctype.h	int isascii(int c);	*c가 7비트 US-ASCII 범위에 포함되는지 테스트합니다.*		
isblank	ctype.h	int isblank(int c);	*c가 공백 또는 탭 문자인지 테스트합니다.*		
iscntrl	ctype.h	int iscntrl(int c);	*c가 제어 문자인지 테스트합니다.*		
isdigit	ctype.h	int isdigit(int c);	*c가 10진수인지 테스트합니다.*		
isgraph	ctype.h	int isgraph(int c);	*c가 공백을 제외한 인쇄 가능한 문자인지 테스트합니다.*		
islower	ctype.h	int islower(int c);	*c가 소문자인지 테스트합니다.*		
isprint	ctype.h	int isprint(int c);	*c가 공백을 포함한 인쇄 가능한 문자인지 테스트합니다.*		
ispunct	ctype.h	int ispunct(int c);	*c가 구두점 문자인지 테스트합니다.*		
isspace	ctype.h	int isspace(int c);	*c가 공백 문자인지 테스트합니다.*		
isupper	ctype.h	int isupper(int c);	*c가 대문자인지 테스트합니다.*		
toascii	ctype.h	int toascii(int c);	*c를 7비트 US-ASCII 문자 세트의 문자로 변환합니다.*		
tolower	ctype.h	int tolower(int c);	*c를 소문자로 변환합니다.*		
toupper	ctype.h	int toupper(int c);	*c를 대문자로 변환합니다.*		
nl_langinfo4	langinfo.h	char *nl_langinfo(nl_item item);	*item에서 지정하는 요청된 정보를 설명하는 스트링을 현재 로케일에서 검색합니다.*		
localeconv	locale.h	struct lconv *localeconv(void);	현재 로케일에 따라 구조 lconv에서 숫자 수량을 형식화합니다.		
setlocale	locale.h	char *setlocale(int category, const char *locale);	*locale에 정의된 변수를 변경 또는 쿼리합니다.*		
wcslocaleconv	locale.h	struct wcslconv *wcslocaleconv(void);	현재 로케일에 따라 struct wcslconv에서 숫자 수량을 형식화합니다.		
acos	math.h	double acos(double x);	*x의 역코사인을 계산합니다.*		
asin	math.h	double asin(double x);	*x의 역사인을 계산합니다.*		
atan	math.h	double atan(double x);	*x의 역탄젠트를 계산합니다.*		
atan2	math.h	double atan2(double y, double x);	*y/x의 역탄젠트를 계산합니다.*		
ceil	math.h	double ceil(double x);	*x 이상의 가장 작은 정수를 나타내는 double 값을 계산합니다.*		
cos	math.h	double cos(double x);	*x의 코사인을 계산합니다.*		
cosh	math.h	double cosh(double x);	*x의 쌍곡 코사인을 계산합니다.*		
erf	math.h	double erf(double x);	*x의 오류 함수를 계산합니다.*		
erfc	math.h	double erfc(double x);	*x의 큰 값에 대한 오류 함수를 계산합니다.*		
exp	math.h	double exp(double x);	부동 소수점 인수 x의 지수 함수를 계산합니다.		
fabs	math.h	double fabs(double x);	부동 소수점 인수 x의 절대값을 계산합니다.		
floor	math.h	double floor(double x);	*x 이하의 가장 큰 정수를 나타내는 부동 소수점 값을 계산합니다.*		
fmod	math.h	double fmod(double x, double y);	*x/y의 부동 소수점 나머지를 계산합니다.*		
frexp	math.h	double frexp(double x, int *expptr);	부동 소수점 숫자를 가수와 지수로 분리합니다.		
gamma	math.h	double gamma(double x);	감마 함수 계산		
hypot	math.h	double hypot(double side1, double side2);	변 길이가 side1 및 side2인 직각 삼각형의 빗변을 계산합니다.		
j0	math.h	double j0(double x);	순서 0의 첫 번째 종류에 대한 베셀 함수 값을 계산합니다.		
j1	math.h	double j1(double x);	순서 1의 첫 번째 종류에 대한 베셀 함수 값을 계산합니다.		
jn	math.h	double jn(int n, double x);	순서 n의 첫 번째 종류에 대한 베셀 함수 값을 계산합니다.		
ldexp	math.h	double ldexp(double x, int exp);	*x에 (2의 exp제곱)을 곱한 값을 리턴합니다.*		
log	math.h	double log(double x);	*x의 자연 로그를 계산합니다.*		
log10	math.h	double log10(double x);	*x의 밑이 10인 로그를 계산합니다.*		
modf	math.h	double modf(double x, double *intptr);	부동 소수점 값 x를 소수 및 정수 파트로 구분합니다.		
nextafter	math.h	double nextafter(double x, double y);	*x 이후 y 방향으로 다음에 표시 가능한 값을 계산합니다.*		
nextafterl	math.h	long double nextafterl(long double x, long double y);	*x 이후 y 방향으로 다음에 표시 가능한 값을 계산합니다.*		
nexttoward	math.h	double nexttoward(double x, long double y);	*x 이후 y 방향으로 다음에 표시 가능한 값을 계산합니다.*		
nexttowardl	math.h	long double nexttowardl(long double x, long double y);	*x 이후 y 방향으로 다음에 표시 가능한 값을 계산합니다.*		
pow	math.h	double pow(double x, double y);	값 x의 y제곱을 계산합니다.		
quantexpd32	math.h	_Decimal32 quantized32(_Decimal32 x, _Decimal32 y);	단정밀도 십진 부동 소수값의 퀀텀 지수를 계산합니다.		
quantexpd64	math.h	_Decimal64 quantized64(_Decimal64 x, _Decimal64 y);	배정밀도 십진 부동 소수값의 퀀텀 지수를 계산합니다.		
quantexpd128	math.h	_Decimal128 quantized128(_Decimal128 x, _Decimal128 y);	4배정밀도 십진 부동 소수값의 퀀텀 지수를 계산합니다.		
quantized32	math.h	int quantexpd32(_Decimal32 x);	단정밀도 십진 부동 소수값의 퀀텀 지수를 다른 단정밀도 십진 부동 소수값의 퀀텀 지수로 설정합니다.		
quantized64	math.h	int quantexpd64(_Decimal64 x);	배정밀도 십진 부동 소수값의 퀀텀 지수를 다른 배정밀도 십진 부동 소수값의 퀀텀 지수로 설정합니다.		
quantized128	math.h	int quantexpd128(_Decimal128 x);	4배정밀도 십진 부동 소수값의 퀀텀 지수를 다른 4배정밀도 십진 부동 소수값의 퀀텀 지수로 설정합니다.		
samequantum	math.h	__bool__ samequantumd32(_Decimal32 x, _Decimal32 y);	두 단정밀도 십진 부동 소수값의 퀀텀 지수가 동일한지 판별합니다.		
samequantum	math.h	__bool__ samequantumd64(_Decimal64 x, _Decimal64 y);	두 배정밀도 십진 부동 소수값의 퀀텀 지수가 동일한지 판별합니다.		
samequantum	math.h	__bool__ samequantumd128(_Decimal128 x, _Decimal128 y);	두 4배정밀도 십진 부동 소수값의 퀀텀 지수가 동일한지 판별합니다.		
sin	math.h	double sin(double x);	*x의 사인을 계산합니다.*		
sinh	math.h	double sinh(double x);	*x의 쌍곡 사인을 계산합니다.*		
sqrt	math.h	double sqrt(double x);	*x의 제곱근을 계산합니다.*		
tan	math.h	double tan(double x);	*x의 탄젠트를 계산합니다.*		
tanh	math.h	double tanh(double x);	*x의 쌍곡 탄젠트를 계산합니다.*		
y0	math.h	double y0(double x);	순서 0의 두 번째 종류에 대한 베셀 함수 값을 계산합니다.		
y1	math.h	double y1(double x);	순서 1의 두 번째 종류에 대한 베셀 함수 값을 계산합니다.		

yn	math.h	double yn(int n, double x);	순서 n의 두 번째 종류에 대한 베셀 함수 값을 계산합니다.		
catclose6	nl_types.h	int catclose (nl_catd catd);	이전에 열린 메세지 카탈로그를 닫습니다.		
catgets6	nl_types.h	char *catgets(nl_catd catd, int set_id, int msg_id, const char *s);	열린 메세지 카탈로그에서 메세지를 검색합니다.		
catopen6	nl_types.h	nl_catd catopen (const char *name, int oflag);	메세지 카탈로그를 엽니다. 이 작업은 메세지를 검색하기 전에 수행해야 합니다.		
regcomp	regex.h	int regcomp(regex_t *preg, const char *pattern, int cflags);	*pattern으로 가리키는 소스 정규식을 실행 가능 버전으로 컴파일하고 preg로 가리키는 위치에 저장합니다.*		
regerror	regex.h	size_t regerror(int errcode, const regex_t *preg, char *errbuf, size_t errbuf_size);	정규식 preg에 대한 오류 코드 errcode의 설명을 찾습니다.		
regexec	regex.h	int regexec(const regex_t *preg, const char *string, size_t nmatch, regmatch_t *pmatch, int eflags);	널 종료 스트링 string을 컴파일된 정규식 preg와 비교하여 둘 사이의 일치를 찾습니다.		
regfree	regex.h	void regfree(regex_t *preg);	정규식 preg를 구현하도록 regcomp에서 할당한 메모리를 해제합니다.		
longjmp	setjmp.h	void longjmp(jmp_buf env, int value);	setjmp 함수에서 env에 전에 설정된 스택 환경을 복원합니다.		
setjmp	setjmp.h	int setjmp(jmp_buf env);	스택 환경을 저장합니다. 그러면 후속으로 longjmp에서 복원할 수 있습니다.		
raise	signal.h	int raise(int sig);	신호 sig를 실행 중인 프로그램에 전송합니다.		
signal	signal.h	void(*signal (int sig, void(*func)(int))) (int);	신호 sig에 대한 신호 핸들러로 func를 등록합니다.		
va_arg	stdarg.h	*var_type va_arg(va_list arg_ptr, var_type);*	한 인수 값을 리턴하고 다음 인수를 가리키도록 arg_ptr을 수정합니다.		
va_copy	stdarg.h	void va_copy(va_list dest, va_list src);	*src의 사본으로 dest를 초기화합니다.*		
va_end	stdarg.h	void va_end(va_list arg_ptr);	가변 인수 리스트 처리에서 정상 리턴을 사용합니다.		
va_start	stdarg.h	void va_start(va_list arg_ptr, variable_name);	*va_arg 및 va_end에 의한 후속 사용 시 arg_ptr을 초기화합니다.*		
vfwprintf6	stdarg.hstdio.h wchar.h	int vfwprintf(FILE *stream, const wchar_t *format, va_list arg);	가변 인수 리스트가 arg로 대체된다는 점을 제외하고 fwprintf와 동일합니다.		
vswprintf	stdarg.hwchar.h	int vswprintf(wchar_t *wcsbuffer, size_t n, const wchar_t *format, va_list arg);	와이드 문자 및 값의 시리즈를 형식화하고 버퍼 wcsbuffer에 저장합니다.		
vwprintf6	stdarg.hwchar.h	int vwprintf(const wchar_t *format, va_list arg);	가변 인수 리스트가 arg로 대체된다는 점을 제외하고 wprintf와 동일합니다.		
wctob	stdarg.hwchar.h	int wctob(wint_t wc);	초기 시프트 상태인 경우 멀티바이트 문자 표시가 1바이트인 확장된 문자 세트의 멤버에 wc가 대응하는지 여부를 판별합니다.		
clearerr	stdio.h	void clearerr(FILE *stream);	*stream에 대한 오류 인디케이터 및 파일의 끝 인디케이터를 재설정합니다.*		
fclose	stdio.h	int fclose(FILE *stream);	지정된 스트림을 닫습니다.		
fdopen5	stdio.h	FILE *fdopen(int handle, const char *type);	핸들로 식별된 파일과 입력 또는 출력 스트림을 연관합니다.		
feof	stdio.h	int feof(FILE *stream);	파일의 끝 플래그가 지정된 stream에 대해 설정되었는지 테스트합니다.		
ferror	stdio.h	int ferror(FILE *stream);	*stream에서 읽거나 해당 스트림에 기록하는 경우 오류 인디케이터를 테스트합니다.*		
fflush1	stdio.h	int fflush(FILE *stream);	출력 stream과 연관된 버퍼의 컨텐츠를 기록합니다.		
fgetc1	stdio.h	int fgetc(FILE *stream);	입력 stream에서 부호없는 단일 문자를 읽습니다.		
fgetpos1	stdio.h	int fgetpos(FILE *stream, fpos_t *pos);	*stream과 연관된 파일 포인터의 현재 위치를 pos로 가리킨 오브젝트에 저장합니다.*		
fgets1	stdio.h	char *fgets(char *string, int n, FILE *stream);	입력 stream에서 스트링을 읽습니다.		
fileno5	stdio.h	int fileno(FILE *stream);	현재 stream과 연관된 파일 핸들을 판별합니다.		
fopen	stdio.h	FILE *fopen(const char *filename, const char *mode);	지정된 파일을 엽니다.		
fprintf	stdio.h	int fprintf(FILE *stream, const char *format-string, arg-list);	문자 및 값을 형식화하고 출력 stream에 인쇄합니다.		
fputc1	stdio.h	int fputc(int c, FILE *stream);	문자를 출력 stream에 인쇄합니다.		
fputs1	stdio.h	int fputs(const char *string, FILE *stream);	스트링을 출력 stream에 복사합니다.		
fread	stdio.h	size_t fread(void *buffer, size_t size, size_t count, FILE *stream);	입력 stream에서 size 길이의 항목을 최대 count회 읽고 buffer에 저장합니다.		
freopen	stdio.h	FILE *freopen(const char *filename, const char *mode, FILE *stream);	*stream을 닫고 지정된 파일에 재지정합니다.*		
fscanf	stdio.h	int fscanf(FILE *stream, const char *format-string, arg-list);	*stream에서 arg-list로 지정된 위치로 데이터를 읽습니다.*		
fseek1	stdio.h	int fseek(FILE *stream, long int offset, int origin);	*stream과 연관된 현재 파일 위치를 새 위치로 변경합니다.*		
fsetpos1	stdio.h	int fsetpos(FILE *stream, const fpos_t *pos);	현재 파일 위치를 pos로 판별된 새 위치로 이동합니다.		
ftell1	stdio.h	long int ftell(FILE *stream);	파일 포인터의 현재 위치를 가져옵니다.		
fwrite	stdio.h	size_t fwrite(const void *buffer, size_t size,size_t count, FILE *stream);	*buffer에서 stream까지 size 길이의 항목을 최대 count회 기록합니다.*		
getc1	stdio.h	int getc(FILE *stream);	입력 stream에서 단일 문자를 읽습니다.		
getchar1	stdio.h	int getchar(void);	*stdin에서 단일 문자를 읽습니다.*		
gets	stdio.h	char *gets(char *buffer);	*stdin에서 스트링을 읽고, buffer에 저장합니다.*		
perror	stdio.h	void perror(const char *string);	stderr에 오류 메세지를 인쇄합니다.		
printf	stdio.h	int printf(const char *format-string, arg-list);	문자 및 값을 형식화하고 stdout에 인쇄합니다.		
putc1	stdio.h	int putc(int c, FILE *stream);	*c를 출력 stream에 인쇄합니다.*		
putchar1	stdio.h	int putchar(int c);	*c를 stdout에 인쇄합니다.*		
puts	stdio.h	int puts(const char *string);	스트링을 stdout에 인쇄합니다.		
remove	stdio.h	int remove(const char *filename);	*filename으로 지정된 파일을 삭제합니다.*		
rename	stdio.h	int rename(const char *oldname, const char *newname);	지정된 파일명을 변경합니다.		
rewind1	stdio.h	void rewind(FILE *stream);	*stream과 연관된 파일 포인터를 파일 시작 위치로 재배치합니다.*		
scanf	stdio.h	int scanf(const char *format-string, arg-list);	stdin에서 arg-list로 지정된 위치로 데이터를 읽습니다.		
setbuf	stdio.h	void setbuf(FILE *stream, char *buffer);	*stream에 대한 버퍼링을 제어합니다.*		
setvbuf	stdio.h	int setvbuf(FILE *stream, char *buf, int type, size_t size);	*stream에 대한 size 버퍼 및 버퍼링을 제어합니다.*		
snprintf	stdio.h	int snprintf(char *outbuf, size_t n, const char*, ...)	n자를 outbuf에 기록한 후에 함수가 중단된다는 점을 제외하고 sprintf와 동일합니다.		
sprintf	stdio.h	int sprintf(char *buffer, const char *format-string, arg-list);	문자 및 값을 형식화하고 buffer에 저장합니다.		
sscanf	stdio.h	int sscanf(const char *buffer, const char *format, arg-list);	*buffer에서 arg-list로 지정된 위치로 데이터를 읽습니다.*		
tmpfile	stdio.h	FILE *tmpfile(void);	임시 2진 파일을 작성하고 엽니다.		
tmpnam	stdio.h	char *tmpnam(char *string);	임시 파일명을 생성합니다.		
ungetc1	stdio.h	int ungetc(int c, FILE *stream);	*c를 입력 stream에 다시 푸시합니다.*		
vsnprintf	stdio.h	int vsnprintf(char *outbuf, size_t n, const char*, va_list);	n자를 outbuf에 기록한 후에 함수가 중단된다는 점을 제외하고 vsprintf와 동일합니다.		
btowc	stdio.h wchar.h	wint_t btowc(int c);	*c가 초기 시프트 상태에서 유효한 멀티바이트 문자를 구성하는지 여부를 판별합니다.*		
fgetwc6	stdio.h wchar.h	wint_t fgetwc(FILE *stream);	*stream로 가리키는 입력 스트림에서 다음 멀티바이트 문자를 읽습니다.*		

fgetws6	stdio.h wchar.h	wchar_t *fgetws(wchar_t *wcs, int n, FILE *stream);	스트림에서 wcs로 가리키는 배열로 와이드 문자를 읽습니다.		
fputwc6	stdio.h wchar.h	wint_t fputwc(wchar_t wc, FILE *stream);	와이드 문자 wc를 멀티바이트 문자로 변환하고 현재 위치에서 stream으로 가리키는 출력 스트림에 기록합니다.		
fputws6	stdio.h wchar.h	int fputws(const wchar_t *wcs, FILE *stream);	와이드 문자 스트링 wcs를 멀티바이트 문자 스트링으로 변환하고 멀티바이트 문자 스트링으로 stream에 기록합니다.		
fwide6	stdio.h wchar.h	int fwide(FILE *stream, int mode);	*stream으로 가리키는 스트림의 방향을 판별합니다.*		
fwprintf6	stdio.h wchar.h	int fwprintf(FILE *stream, const wchar_t *format, arg-list);	*stream으로 가리키는 스트림에 출력을 기록합니다.*		
fwscanf6	stdio.h wchar.h	int fwscanf(FILE *stream, const wchar_t *format, arg-list)	*stream으로 가리키는 스트림에서 입력을 읽습니다.*		
getwc6	stdio.h wchar.h	wint_t getwc(FILE *stream);	*stream에서 다음 멀티바이트 문자를 읽고 와이드 문자로 변환한 후 stream에 대해 연관된 파일 위치 인디케이터를 진행합니다.*		
putwc6	stdio.h wchar.h	wint_t putwchar(wchar_t wc, FILE *stream);	와이드 문자 wc를 멀티바이트 문자로 변환하고 현재 위치에서 스트림에 기록합니다.		
ungetwc6	stdio.h wchar.h	wint_t ungetwc(wint_t wc, FILE *stream);	와이드 문자 wc를 입력 스트림에 다시 푸시합니다.		
vfprintf	stdio.h stdarg.h	int vfprintf(FILE *stream, const char *format, va_list arg_ptr);	문자를 형식화하고 가변 개수의 인수를 사용하여 출력 stream에 문자를 인쇄합니다.		
vfscanf	stdio.h stdarg.h	int vfscanf(FILE *stream, const char *format, va_list arg_ptr);	지정된 스트림에서 가변 개수의 인수로 지정된 위치로 데이터를 읽습니다.		
vfwscanf	stdio.h stdarg.h	int vfwscanf(FILE *stream, const wchar_t *format, va_list arg_ptr);	지정된 스트림에서 가변 개수의 인수로 지정된 위치로 와이드 데이터를 읽습니다.		
vprintf	stdio.h stdarg.h	int vprintf(const char *format, va_list arg_ptr);	가변 개수의 인수를 사용하여 문자를 형식화하고 stdout에 인쇄합니다.		
vscanf	stdio.h stdarg.h	int vscanf(const char *format, va_list arg_ptr);	stdin에서 가변 개수의 인수로 지정된 위치로 데이터를 읽습니다.		
vsprintf	stdio.h stdarg.h	int vsprintf(char *target-string, const char *format, va_list arg_ptr);	가변 개수의 인수를 사용하여 문자를 형식화하고 버퍼에 저장합니다.		
vsscanf	stdio.h stdarg.h	int vsscanf(const char*buffer, const char *format, va_list arg_ptr);	버퍼에서 가변 개수의 인수로 지정된 위치로 데이터를 읽습니다.		
vswscanf	stdio.h wchar.h	int vswscanf(const wchar_t *buffer, const wchar_t *format, va_list arg_ptr);	버퍼에서 가변 개수의 인수로 지정된 위치로 와이드 데이터를 읽습니다.		
vwscanf	stdio.h wchar.h	int vwscanf(const wchar_t *format, va_list arg_ptr);	stdin에서 가변 개수의 인수로 지정된 위치로 와이드 데이터를 읽습니다.		
abort	stdlib.h	void abort(void);	비정상적으로 프로그램을 중지합니다.		
abs	stdlib.h	int abs(int n);	정수 인수 n의 절대값을 계산합니다.		
atexit	stdlib.h	int atexit(void (*func)(void));	정상 종료 시 호출할 함수를 등록합니다.		
atof	stdlib.h	double atof(const char *string);	*string을 배정밀도 부동 소수점 값으로 변환합니다.*		
atoi	stdlib.h	int atoi(const char *string);	*string을 정수로 변환합니다.*		
atol	stdlib.h	long int atol(const char *string);	*string을 long integer로 변환합니다.*		
bsearch	stdlib.h	void *bsearch(const void *key, const void *base, size_t num, size_t size, int (*compare) (const void *element1, const void *element2));	*num 요소의 배열에서 2진 검색을 수행합니다(각각 size바이트). 배열은 compare로 가리키는 함수에서 오름차순으로 저장되어야 합니다.*		
calloc	stdlib.h	void *calloc(size_t num, size_t size);	*num 요소의 배열에 대한 기억장치 공간을 예약하고(각각 size바이트) 모든 요소 값을 0으로 초기화합니다.*		
div	stdlib.h	div_t div(int numerator, int denominator);	*numerator를 denominator로 나눈 경우 몫과 나머지를 계산합니다.*		
exit	stdlib.h	void exit(int status);	정상적으로 프로그램을 종료합니다.		
free	stdlib.h	void free(void *ptr);	기억장치 블록을 해제합니다.		
getenv	stdlib.h	char *getenv(const char *varname);	*varname에 대한 환경 변수를 검색합니다.*		
labs	stdlib.h	long int labs(long int n);	*n의 절대값을 계산합니다.*		
ldiv	stdlib.h	ldiv_t ldiv(long int numerator, long int denominator);	*numerator/denominator 연산의 몫과 나머지를 계산합니다.*		
malloc	stdlib.h	void *malloc(size_t size);	기억장치 블록을 예약합니다.		
mblen	stdlib.h	int mblen(const char *string, size_t n);	멀티바이트 문자 string의 길이를 판별합니다.		
mbstowcs	stdlib.h	size_t mbstowcs(wchar_t *pwc, const char *string, size_t n);	*string의 멀티바이트 문자를 대응하는 wchar_t 코드로 변환하고 pwc에 n개 이하의 코드를 지정합니다.*		
mbtowc	stdlib.h	int mbtowc(wchar_t *pwc, const char *string, size_t n);	멀티바이트 문자 string의 처음 n바이트에 대응하는 wchar_t 코드를 wchar_t 문자 pwc에 저장합니다.		
putenv	stdlib.h	int *putenv(const char *varname);	기존 변수를 대체하거나 새로 작성하여 환경 변수 값을 설정합니다.		
qsort	stdlib.h	void qsort(void *base, size_t num, size_t width, int(*compare)(const void *element1, const void *element2));	*num 요소의 배열을 빠르게 정렬합니다(각각 크기가 width바이트).*		
rand	stdlib.h	int rand(void);	의사 난수 정수를 리턴합니다.		
rand_r	stdlib.h	int rand_r(void);	의사 난수 정수를 리턴합니다. (재시작 가능 버전)		
realloc	stdlib.h	void *realloc(void *ptr, size_t size);	이전에 예약된 기억장치 블록의 size를 변경합니다.		
srand	stdlib.h	void srand(unsigned int seed);	의사 난수 생성기에 대한 seed를 설정합니다.		
strtod	stdlib.h	double strtod(const char *nptr, char **endptr);	*nptr을 배정밀도 값으로 변환합니다.*		
strtod32	stdlib.h	_Decimal32 strtod32(const char *nptr, char **endptr);	*nptr을 단정밀도 십진 부동 소수값으로 변환합니다.*		
strtod64	stdlib.h	_Decimal64 strtod64(const char *nptr, char **endptr);	*nptr을 배정밀도 십진 부동 소수값으로 변환합니다.*		
strtod128	stdlib.h	_Decimal128 strtod128(const char *nptr, char **endptr);	*nptr을 4배정밀도 십진 부동 소수값으로 변환합니다.*		
strtof	stdlib.h	float strtof(const char *nptr, char **endptr);	*nptr을 부동 값으로 변환합니다.*		
strtol	stdlib.h	long int strtol(const char *nptr, char **endptr, int base);	*nptr을 부호있는 long integer로 변환합니다.*		
strtold	stdlib.h	long double strtold(const char *nptr, char **endptr);	*nptr을 long double 값으로 변환합니다.*		
strtoul	stdlib.h	unsigned long int strtoul(const char *string1, char **string2, int base);	*string1을 부호없는 long integer로 변환합니다.*		
시스템	stdlib.h	int system(const char *string);	*string을 시스템 명령 분석기로 전달합니다.*		
wcstombs	stdlib.h	size_t wcstombs(char *dest, const wchar_t *string, size_t count);	wchar_t string를 멀티바이트 스트링 dest로 변환합니다.		
wctomb	stdlib.h	int wctomb(char *string, wchar_t character);	*character의 wchar_t 값을 멀티바이트 string으로 변환합니다.*		
memchr	string.h	void *memchr(const void *buf, int c, size_t count);	부호없는 문자로 변환된 c의 첫 번째 표시를 buf의 처음 count바이트에서 검색합니다.		
memcmp	string.h	int memcmp(const void *buf1, const void *buf2, size_t count);	*buf1 및 buf2의 최대 count바이트를 비교합니다.*		
memcpy	string.h	void *memcpy(void *dest, const void *src, size_t count);	*src의 count바이트를 dest에 복사합니다.*		
memmove	string.h	void *memmove(void *dest, const void *src, size_t count);	*src의 count바이트를 dest에 복사합니다. 겹치는 오브젝트 간 복사가 허용됩니다.*		
memset	string.h	void *memset(void *dest, int c, size_t count);	*dest에서 count바이트를 값 c로 설정합니다.*		
strcat	string.h	char *strcat(char *string1, const char *string2);	*string2를 string1에 연결합니다.*		
strchr	string.h	char *strchr(const char *string, int c);	*string에서 c의 첫 번째 표시를 찾습니다.*		
strcmp	string.h	int strcmp(const char *string1, const char *string2);	*string1의 값을 string2와 비교합니다.*		

strcoll	string.h	int strcoll(const char *string1, const char *string2);	현재 로케일에서 배열 순서를 사용하여 두 스트링을 비교합니다.		
strcpy	string.h	char *strcpy(char *string1, const char *string2);	*string2*를 *string1*에 복사합니다.		
strcspn	string.h	size_t strcspn(const char *string1, const char *string2);	*string2*에 포함되지 않는 문자로 구성된 *string1*의 초기 서브스트링 길이를 리턴합니다.		
strerror	string.h	char *strerror(int errnum);	오류 메세지 스트링에 errnum의 오류 번호를 맵핑합니다.		
strlen	string.h	size_t strlen(const char *string);	*string* 길이를 계산합니다.		
strncat	string.h	char *strncat(char *string1, const char *string2, size_t count);	*string2*에서 최대 *count*자를 *string1*에 연결합니다.		
strncmp	string.h	int strncmp(const char *string1, const char *string2, size_t count);	*string1* 및 *string2*의 최대 *count*자를 비교합니다.		
strncpy	string.h	char *strncpy(char *string1, const char *string2, size_t count);	*string2*에서 최대 *count*자를 *string1*에 복사합니다.		
strpbrk	string.h	char *strpbrk(const char *string1, const char *string2);	*string2*에 있는 임의 문자의 첫 번째 표시를 *string1*에서 찾습니다.		
strrchr	string.h	char *strrchr(const char *string, int c);	*string*에서 *c*의 마지막 표시를 찾습니다.		
strspn	string.h	size_t strspn(const char *string1, const char *string2);	*string2*에 포함된 문자로 구성된 *string1*의 초기 서브스트링 길이를 리턴합니다.		
strstr	string.h	char *strstr(const char *string1, const char *string2);	*string1*에서 *string2*의 첫 번째 표시에 대한 포인터를 리턴합니다.		
strtok	string.h	char *strtok(char *string1, const char *string2);	*string2*에서 다음 문자로 구분된 *string1*의 다음 토큰을 찾습니다.		
strtok_r	string.h	char *strtok_r(char *string, const char *seps, char **lasts);	*seps*에서 다음 문자로 구분된 *string*의 다음 토큰을 찾습니다. *(strtok의 재시작 가능 버전.)*		
strxfrm	string.h	size_t strxfrm(char *string1, const char *string2, size_t count);	*string2*를 변환하고 *string1*에 결과를 배치합니다. 변환은 프로그램의 현재 로케일에 의해 판별됩니다.		
strcasecmp	strings.h	int strcasecmp(const char *string1, const char *string2);	대소문자를 구분하지 않고 스트링을 비교합니다.		
strncasecmp	strings.h	int strncasecmp(const char *string1, const char *string2, size_t count);	대소문자를 구분하지 않고 스트링을 비교합니다.		
asctime	time.h	char *asctime(const struct tm *time);	구조로 저장된 time을 문자 스트링으로 변환합니다.		
asctime_r	time.h	char *asctime_r (const struct tm *tm, char *buf);	구조로 저장된 tm을 문자 스트링으로 변환합니다. (asctime의 재시작 가능 버전.)		
clock	time.h	clock_t clock(void);	작업을 시작한 이후 경과한 프로세서 시간을 리턴합니다.		
ctime	time.h	char *ctime(const time_t *time);	*time*을 문자 스트링으로 변환합니다.		
ctime64	time.h	char *ctime64(const time64_t *time);	*time*을 문자 스트링으로 변환합니다.		
ctime_r	time.h	char *ctime_r(const time_t *time, char *buf);	*time*을 문자 스트링으로 변환합니다. *(ctime의 재시작 가능 버전.)*		
ctime64_r	time.h	char *ctime64_r(const time64_t *time, char *buf);	*time*을 문자 스트링으로 변환합니다. *(ctime64의 재시작 가능 버전.)*		
difftime	time.h	double difftime(time_t time2, time_t time1);	*time2* 및 *time1* 사이의 차이를 계산합니다.		
difftime64	time.h	double difftime64(time64_t time2, time64_t time1);	*time2* 및 *time1* 사이의 차이를 계산합니다.		
gmtime	time.h	struct tm *gmtime(const time_t *time);	*time* 값을 *tm* 유형의 구조로 변환합니다.		
gmtime64	time.h	struct tm *gmtime64(const time64_t *time);	*time* 값을 *tm* 유형의 구조로 변환합니다.		
gmtime_r	time.h	struct tm *gmtime_r (const time_t *time, struct tm *result);	*time* 값을 *tm* 유형의 구조로 변환합니다. *(gmtime의 재시작 가능 버전.)*		
gmtime64_r	time.h	struct tm *gmtime64_r (const time64_t *time, struct tm *result);	*time* 값을 *tm* 유형의 구조로 변환합니다. *(gmtime64의 재시작 가능 버전.)*		
localtime	time.h	struct tm *localtime(const time_t *timeval);	*timeval*을 유형 *tm*의 구조로 변환합니다.		
localtime64	time.h	struct tm *localtime64(const time64_t *timeval);	*timeval*을 유형 *tm*의 구조로 변환합니다.		
localtime_r	time.h	struct tm *localtime_r (const time_t *timeval, struct tm *result);	*time* 값을 유형 *tm*의 구조로 변환합니다. *(localtime의 재시작 가능 버전.)*		
localtime64_r	time.h	struct tm *localtime64_r (const time64_t *timeval, struct tm *result);	*time* 값을 유형 *tm*의 구조로 변환합니다. *(localtime64의 재시작 가능 버전.)*		
mktime	time.h	time_t mktime(struct tm *time);	로컬 time을 캘린더 시간으로 변환합니다.		
mktime64	time.h	time64_t mktime64(struct tm *time);	로컬 time을 캘린더 시간으로 변환합니다.		
strftime	time.h	size_t strftime (char *dest, size_t maxsize, const char *format, const struct tm *timeptr);	*format*으로 판별된 스트링에 따라, *dest*로 가리키는 배열에 문자를 저장합니다.		
strptime4	time.h	char *strptime(const char *buf, const char *format, struct tm *tm);	날짜 및 시간 변환		
시간	time.h	time_t time(time_t *timeptr);	현재 캘린더 시간을 리턴합니다.		
time64	time.h	time64_t time64(time64_t *timeptr);	현재 캘린더 시간을 리턴합니다.		
getwchar6	wchar.h	wint_t getwchar(void);	stdin에서 다음 멀티바이트 문자를 읽고 와이드 문자로 변환한 후 stdin에 대해 연관된 파일 위치 인디케이터를 진행합니다.		
mbrlen4	wchar.h	int mbrlen (const char *s, size_t n, mbstate_t *ps);	멀티바이트 문자의 길이를 판별합니다. (mblen의 재시작 가능 버전.)		
mbrtowc4	wchar.h	int mbrtowc (wchar_t *pwc, const char *s, size_t n, mbstate_t *ps);	멀티바이트 문자를 와이드 문자로 변환합니다 (mbtowc의 재시작 가능 버전).		
mbsinit4	wchar.h	int mbsinit (const mbstate_t *ps);	상태 오브젝트 *ps에서 초기 상태를 테스트합니다.		
mbsrtowcs4	wchar.h	size_t mbsrtowc (wchar_t *dst, const char **src, size_t len, mbstate_t *ps);	멀티바이트 스트링을 와이드 문자 스트링으로 변환합니다. (mbstowcs의 재시작 가능 버전.)		
putwchar6	wchar.h	wint_t putwchar(wchar_t wc);	와이드 문자 wc를 멀티바이트 문자로 변환하고 stdout에 기록합니다.		
strfmon4	wchar.h	int strfmon (char *s, size_t maxsize, const char *format, ...);	통화 값을 스트링으로 변환합니다.		
swprintf	wchar.h	int swprintf(wchar_t *wcsbuffer, size_t n, const wchar_t *format, arg-list);	와이드 문자 및 값의 시리즈를 형식화하고 와이드 문자 버퍼 wcsbuffer에 저장합니다.		
swscanf	wchar.h	int swscanf (const wchar_t *buffer, const wchar_t *format, arg-list)	*buffer*에서 *arg-list*로 지정한 위치로 데이터를 읽습니다.		
wcrtomb4	wchar.h	int wcrtomb (char *s, wchar_t wchar, mbstate_t *pss);	와이드 문자를 멀티바이트 문자로 변환합니다. (wctomb의 재시작 가능 버전.)		
wcscat	wchar.h	wchar_t *wcscat(wchar_t *string1, const wchar_t *string2);	*string2*로 가리키는 스트링의 사본을 *string1*로 가리키는 스트링의 끝에 추가합니다.		
wcschr	wchar.h	wchar_t *wcschr(const wchar_t *string, wchar_t character);	*string*으로 가리키는 와이드 문자 스트링에서 *character*의 표시를 검색합니다.		
wcscmp	wchar.h	int wcscmp(const wchar_t *string1, const wchar_t *string2);	두 와이드 문자 스트링, *string1 및 *string2를 비교합니다.		
wcscoll4	wchar.h	int wcscoll (const wchar_t *wcs1, const wchar_t *wcs2);	현재 로케일에서 배열 순서를 사용하여 두 와이드 문자 스트링을 비교합니다.		
wcscpy	wchar.h	wchar_t *wcscpy(wchar_t *string1, const wchar_t *string2);	**string2*의 컨텐츠(종료 *wchar_t* 널 문자 포함)를 **string1*로 복사합니다.		
wcscspn	wchar.h	size_t wcscspn(const wchar_t *string1, const wchar_t *string2);	**string1*로 가리키는 스트링의 초기 세그먼트에서 **string2*로 가리키는 스트링에 나타나지 않는 *wchar_t* 문자 수를 판별합니다.		
wcsftime	wchar.h	size_t wcsftime(wchar_t *wdest, size_t maxsize, const wchar_t *format, const struct tm *timeptr);	*timeptr* 구조의 시간 및 날짜 스펙을 와이드 문자 스트링으로 변환합니다.		
wcslen	wchar.h	size_t wcslen(const wchar_t *string);	*string*으로 가리키는 스트링에서 와이드 문자 수를 계산합니다.		
wcsncat	wchar.h	wchar_t *wcsncat(wchar_t *string1, const wchar_t *string2, size_t count);	*string2*의 최대 *count*개 와이드 문자를 *string1*의 끝에 추가하고 결과에 *wchar_t* 널 문자를 추가합니다.		
wcsncmp	wchar.h	int wcsncmp(const wchar_t *string1, const wchar_t *string2, size_t count);	*string1*의 최대 *count*개 와이드 문자를 *string2*와 비교합니다.		
wcsncpy	wchar.h	wchar_t *wcsncpy(wchar_t *string1, const wchar_t *string2, size_t count);	*string2*에서 최대 *count*개 와이드 문자를 *string1*에 복사합니다.		
wcspbrk	wchar.h	wchar_t *wcspbrk(const wchar_t *string1, const wchar_t *string2);	*string2*로 가리키는 스트링에서 *string1*로 가리키는 스트링에 있는 와이드 문자의 첫 번째 표시를 찾습니다.		
wcsptime	wchar.h	wchar_t *wcsptime(const wchar_t *buf, const wchar_t *format, struct tm *tm);	날짜 및 시간 변환. 와이드 문자를 사용한다는 점을 제외하고 strptime()과 동일합니다.		
wcsrchr	wchar.h	wchar_t *wcsrchr(const wchar_t *string, wchar_t character);	*string*으로 가리키는 스트링에서 *character*의 마지막 표시를 찾습니다.		
wcsrtombs4	wchar.h	size_t wcsrtombs (char *dst, const wchar_t **src, size_t len, mbstate_t *ps);	와이드 문자 스트링을 멀티바이트 스트링으로 변환합니다. (wcstombs의 재시작 가능 버전.)		

wcsspn	wchar.h	size_t wcsspn(const wchar_t *string1, const wchar_t *string2);	*string1*로 가리키는 스트링의 초기 세그먼트에서 와이드 문자 개수를 계산합니다. 이 세그먼트는 *string2*로 가리키는 스트링에서 완전히 와이드 문자로 구성됩니다.		
wcsstr	wchar.h	wchar_t *wcsstr(const wchar_t *wcs1, const wchar_t *wcs2);	*wcs1*에서 *wcs2*의 첫 번째 표시를 찾습니다.		
wcstod	wchar.h	double wcstod(const wchar_t *nptr, wchar_t **endptr);	*nptr*로 가리키는 와이드 문자 스트링의 초기 부분을 *double* 값으로 변환합니다.		
wcstod32	wchar.h	_Decimal32 wcstod32(const wchar_t *nptr, wchar_t **endptr);	*nptr*로 가리키는 와이드 문자 스트링의 초기 부분을 단정밀도 십진 부동 소수값으로 변환합니다.		
wcstod64	wchar.h	_Decimal64 wcstod64(const wchar_t *nptr, wchar_t **endptr);	*nptr*로 가리키는 와이드 문자 스트링의 초기 부분을 배정밀도 십진 부동 소수값으로 변환합니다.		
wcstod128	wchar.h	_Decimal128 wcstod128(const wchar_t *nptr, wchar_t **endptr);	*nptr*로 가리키는 와이드 문자 스트링의 초기 부분을 4배 정밀도 십진 부동 소수값으로 변환합니다.		
wcstof	wchar.h	float wcstof(const wchar_t *nptr, wchar_t **endptr);	*nptr*로 가리키는 와이드 문자 스트링의 초기 부분을 *float* 값으로 변환합니다.		
wcstok	wchar.h	wchar_t *wcstok(wchar_t *wcs1, const wchar_t *wcs2, wchar_t **ptr)	*wcs1*을 토큰 순서로 구분합니다. 각각 *wcs2*로 가리키는 와이드 스트링의 와이드 문자로 구분합니다.		
wcstol	wchar.h	long int wcstol(const wchar_t *nptr, wchar_t **endptr, int base);	*nptr*로 가리키는 와이드 문자 스트링의 초기 부분을 *long integer* 값으로 변환합니다.		
wcstold	wchar.h	long double wcstold(const wchar_t *nptr, wchar_t **endptr);	*nptr*로 가리키는 와이드 문자 스트링의 초기 부분을 *long double* 값으로 변환합니다.		
wcstoul	wchar.h	unsigned long int wcstoul(const wchar_t *nptr, wchar_t **endptr, int base);	*nptr*로 가리키는 와이드 문자 스트링의 초기 부분을 부호없는 *long integer* 값으로 변환합니다.		
wcsxfrm4	wchar.h	size_t wcsxfrm (wchar_t *wcs1, const wchar_t *wcs2, size_t n);	문자 배열 가중치를 나타내는 값으로 와이드 문자 스트링을 변환하고 결과 와이드 문자 스트링을 배열에 배치합니다.		
wctype4	wchar.h	wctype_t wctype (const char *property);	문자 특성 분류에 대한 핸들을 가져옵니다.		
wcwidth	wchar.h	int wcswidth(const wchar_t *pwcs, size_t n);	와이드 문자 스트링의 표시 너비를 판별합니다.		
wmemchr	wchar.h	wchar_t *wmemchr(const wchar_t *s, wchar_t c, size_t n);	*s*에서 가리킨 오브젝트의 처음 *n*개 와이드 문자에서 *c*의 첫 번째 표시를 찾습니다.		
wmemcmp	wchar.h	int wmemcmp(const wchar_t *s1, const wchar_t *s2, size_t n);	*s1*에서 가리킨 오브젝트의 처음 *n*개 문자를 *s2*에서 가리킨 오브젝트의 처음 *n*개 와이드 문자와 비교합니다.		
wmemcpy	wchar.h	wchar_t *wmemcpy(wchar_t *s1, const wchar_t *s2, size_t n);	*n*개의 와이드 문자를 *s2*에서 가리킨 오브젝트에서 *s1*에서 가리킨 오브젝트로 복사합니다.		
wmemmove	wchar.h	wchar_t *wmemmove(wchar_t *s1, const wchar_t *s2, size_t n);	*n*개의 와이드 문자를 *s2*에서 가리킨 오브젝트에서 *s1*에서 가리킨 오브젝트로 복사합니다.		
wmemset	wchar.h	wchar_t *wmemset(wchar_t *s, wchar_t c, size_t n);	*s*에서 가리킨 오브젝트의 처음 *n*개 와이드 문자에 *c*의 값을 복사합니다.		
wprintf6	wchar.h	int wprintf(const wchar_t *format, arg-list);	wprintf에 대한 인수 이전에 삽입된 인수 stdout를 포함하는 fwprintf와 동일합니다.		
wscanf6	wchar.h	int wscanf(const wchar_t *format, arg-list);	wscanf의 인수 이전에 삽입된 인수 stdin을 포함하는 fwscanf와 동일합니다.		
iswalnum4	wctype.h	int iswalnum (wint_t wc);	영숫자 와이드 문자가 있는지 확인합니다.		
iswalpha4	wctype.h	int iswalpha (wint_t wc);	영문자 와이드 문자가 있는지 확인합니다.		
iswblank4	wctype.h	int iswblank (wint_t wc);	공백 또는 탭 와이드 문자가 있는지 확인합니다.		
iswcntrl4	wctype.h	int iswcntrl (wint_t wc);	제어 와이드 문자를 테스트합니다.		
iswctype4	wctype.h	int iswctype(wint_t wc, wctype_t wc_prop);	와이드 문자 wc에 특성 wc_prop가 있는지 여부를 판별합니다.		
iswdigit4	wctype.h	int iswdigit (wint_t wc);	10진수 와이드 문자가 있는지 확인합니다.		
iswgraph4	wctype.h	int iswgraph (wint_t wc);	와이드 문자 공백을 제외하고 인쇄 와이드 문자가 있는지 확인합니다.		
iswlower4	wctype.h	int iswlower (wint_t wc);	소문자 와이드 문자가 있는지 확인합니다.		
iswprint4	wctype.h	int iswprint (wint_t wc);	인쇄 와이드 문자가 있는지 확인합니다.		
iswpunct4	wctype.h	int iswpunct (wint_t wc);	영숫자, 공백 문자가 아닌 와이드 문자를 테스트합니다.		
iswspace4	wctype.h	int iswspace (wint_t wc);	iswalnum이 false인 와이드 문자의 구현 정의 세트에 대응하는 와이드 문자가 있는지 확인합니다.		
iswupper4	wctype.h	int iswupper (wint_t wc);	대문자 와이드 문자가 있는지 확인합니다.		
iswxdigit4	wctype.h	int iswxdigit (wint_t wc);	16진수 문자가 있는지 확인합니다.		
isxdigit4	wctype.h	int isxdigit(int c);	*c*가 *16*진수인지 테스트합니다.		
towctrans	wctype.h	wint_t towctrans(wint_t wc, wctrans_t desc);	*desc*에서 설명하는 맵핑에 기반하여 와이드 문자 *wc*를 변환합니다.		
towlower4	wctype.h	wint_t towlower(wint_t wc);	대문자를 소문자로 변환합니다.		
towupper4	wctype.h	wint_t towupper(wint_t wc);	소문자를 대문자로 변환합니다.		
wctrans	wctype.h	wctrans_t wctrans(const char *property);	스트링 인수 특성으로 식별된 와이드 문자 사이에서 맵핑을 설명하는 wctrans_t 유형의 값을 구성합니다.		