쌇main
#C-박사 목록

`헤더파일`　　- 헤더파일별 분류
`전역변수`　　- 헤더파일에 정의된 전역변수
`함수`　　- 알파벳순 함수
`기능별함수`　　- 기능별로 분류된 함수
`명령어라인 옵션` - TCC 명령어라인 옵션

쌇헤더파일
#헤더파일

`alloc.h`　　`assert.h`　　`bios.h`
`conio.h`　　`ctype.h`　　`dir.h`
`dos.h`　　`errno.h`　　`fcnl.h`
`float.h`　　`graphics.h`　　`io.h`
`limits.h`　　`math.h`　　`mem.h`
`process.h`　　`setjmp.h`　　`share.h`
`signal.h`　　`stdarg.h`　　`stddef.h`
`stdio.h`　　`stdlib.h`　　`string.h`
`sys\stat.h`　　`sys\timeb.h`　　`time.h`
`values.h`

쌇alloc.h
#헤더파일 ALLOC.H

함수

`brk`　　`calloc`
`coreleft`　　`farcalloc`
`farcoreleft`　　`farfree`
`farmalloc`　　`farrealloc`
`free`　　`malloc`
`realloc`　　`sbrk`

상수, 데이타 타입, 전역변수

`NULL`　　`size_t`

쌇brk
#함수 BRK

기 능 데이타 세그먼트의 할당 영역을 변경한다.
구 문 int brk(void *addr);
프로토타입 `alloc.h`
기능 설명 호출 프로그램의 데이타 세그먼트에 할당된 영역크기를 동적으로 변경하는 데 사용된다. 변경은 프로그램의 break value를 재설정함으로 수행되는데 이는 데이타 세그먼트의 끝을 지나 첫번째 위치의 어드레스가 된다. 할당된 영역의 크기는 break 값이 증가함에 따라 증가한다.

brk는 break 값을 addr 에 설정하고 그에 따라 할당된 영역을 조절한다. 이때 변경이 허용 된 것보다 많은 영역을 요구하는 경우 할당된 공간의 변경을 하지 않으면, 이 함수는 제대로 동작하지 않는다.
리 턴 값 에러없이 수행되면 0을 리턴하고 에러가 발생되면 -1의 값을 리턴한다.
이때 에러의 종류는`errno`으로 판단할수 있다.
이 식 성 UNIX 시스템 에서만 사용할수 있다.
참 조 `coreleft`,`sbrk`

쌇calloc
#함수 CALLOC

기 능 메인 메모리를 할당한다.
구 문 #include<stdlib.h>
void *calloc(size_t nitems, size_t size);
프로토타입 `stdlib.h`,`alloc.h`
기능 설명 c 메모리 heap 에 억세스를 제공한다. heap 은 가변크기 블럭의 동적 할당에 유용하다. 트리나 리스트 같은 데이타 구조는 heap 메모리 할당을 사용한다.

데이타 세그먼트의 맨 끝과 프로그램 스택의 맨 위 바로 전의 256바이트 한계를 제외하고는 작은 메모리 모델 (tiny,small,medium)에서 유용하게 사용된다.

이 한계는 어플리케이션 프로그램을 위해 스택이 증대될 수 있도록 하는 공간과 여기에 DOS 가 필요로 하는 공간을 더한 부분이다. large 데이타 모델(compact,large,huge) 에서는 프로그램이 스택을 넘어 맨마지막의 물리적 영역까지 모든 영역이 heap을 위해 사용 가능한 영역이 된다.

calloc 은 (nitem*size) 크기의 블록을 할당한다. 블록은 0으로 초기화된다.

리 턴 값 새로 할당된 블럭을 가르키는 포인터를 리턴한다. 새 블럭을 할당할 충분한 공간이 없는 경우 또는 nitem 혹은 size 가 0일때 calloc 은`NULL`을 리턴한다.

이 식 성 UNIX 시스템에서 사용될 수 있으며 ANSI-C와 도호환된다.

참 조 `farcalloc`,`free`,`malloc`,`realloc`

쌿coreleft

#함수 CORELEFT

기 능 사용하지 않은 RAM 의 크기를 나타낸다.

구 문 작은 메모리 모델(tiny,small,medium)

```
unsigned coreleft(void);
```

큰 메모리 모델

```
unsigned long coreleft(void);
```

프로토타입 `alloc.h`

기능 설명 사용하지 않은 RAM 의 크기를 리턴하는데 메모리 모델이 small 데이타 그룹인가, large 데이타 그룹인가에 따라서 다른 측정값을 나타낸다.

리 턴 값 large 데이타 모델에서는 coreleft는 heap과 스택사이의 사용하지 않은 메모리의 양을 나타낸다.

이 식 성 DOS 에만 한정된다.

참 조 `allocmem`,`brk`,`farcoreleft`,`malloc`

쌿farcalloc

#함수 FARCALLOC

기 능 far heap 으로부터 메모리를 할당한다.
구 문 void far *farcalloc(unsigned long nunits,
unsigned long unitsz);
프로토타입 `alloc.h`
기능 설명 각 요소가 nunits 바이트 길이로 unitsz 요소의 수만큼 갖는 매열을 위해 far heap으로 부터 메모리를 할당한다. far heap 으로부터 메모리를 할당하면

* 모든 사용 가능한 RAM 을 할당할 수 있다.
* 64K 이상의 메모리 블록을 할당할 수 있다.
* Far 포인터로 블록을 억세스 한다.

compact, large, huge 모델에서는 farcalloc이 calloc 과 비슷한 기능을 갖는다. calloc은 unsigned 형 파라미터를 갖지만 farcalloc은 unsigned long 형 파라미터를 갖는다.

tiny 모델에서는 farcalloc이 .COM 파일로 변환 되어야만 사용할 수 있다.
리 턴 값 새로이 할당된 블록에 대한 포인터를 리턴한다. 새로운 블록을 위한 공간이 부족할 경우에는 NULL 이 리턴된다.
이 식 성 DOS에 한한다.
참 조 `calloc`,`farcoreleft`,`farfree`,`malloc`

쌇farcoreleft
#함수 FARCORELEFT

기 능 far heap 의 사용하지 않은 메모리의 양을 리턴한다.
구 문 unsigned long farcoreleft(void);
프로토타입 `alloc.h`
기능 설명 가장 최상위로 할당됭 블록을 넘어 far heap에서 사용되지 않은 메모리의 양을 리턴한다. tiny 모델에서는 .COM 파일로 변환되지 않으면 사용할 수 없다.
리 턴 값 할당된 메모리의 최상위 블록과 메모리의 맨 마지막 사이에 있는 공간의 총량을 리턴한다.
이 식 성 DOS에 한한다.

참　　조　`coreleft`,`farcalloc`,`farmalloc`

쌇farfree
#함수 FARFREE

기　　능　far heap 으로부터 블록을 해제한다.
구　　문　void farfree(void far *block);
프로토타입　`alloc.h`
기능 설명　far포인터 block 이 가르키는 far 메모리 상
의 영역을 해제하는 함수이다. tiny 모델에서
는 .COM 파일로 변환되어 있지 않으면 사용될
수 없다.

또한 small 모델과 medium 모델 에서는 farma
lloc에 의해 할당된 블록이 정상적인 free 를
통하여는 해제되지 못하며 malloc 을 통해 할
당된 블록도 farfree 에 의해서 해제될 수 없
다.
리 턴 값　없음
이 식 성　DOS 에 한한다.
참　　조　`farcalloc`,`farmalloc`

쌇farmalloc
#함수 FARMALLOC

기　　능　far heap 으로부터 메모리 영역을 할당한다.
구　　문　void far *farmalloc(unsigned long nbytes);
프로토타입　`alloc.h`
기능 설명　far heap 으로부터 nbytes 길이만큼 메모리의
블록을 할당한다.
far heap에서 메모리를 할당하면,

* 모든 사용 가능한 RAM 을 할당할 수 있다.
* 64K 이상의 메모리 블록을 할당할 수 있다.
* Far 포인터로 블록을 억세스 한다.

compact, large 그리고 huge 메모리 모델에서
는 famalloc 은 malloc과 비슷한 기능을 갖는
다. malloc 은 unsigned 파라미터를 갖지만 ,
farmalloc 은 unsigned long 파라미터를 갖는
다. tiny 모델 에서는 COM 파일로 변환되어야

사용 할 수 있다.

리 턴 값 새로이 할당되는 블록 포인터를 리턴한다. 새로운 블록을 위한 공간이 부족하면 NULL이 리턴된다.

이 식 성 DOS에 한한다.

참 조 `farcalloc`,`farcoreleft`,`farfree`,`farrealloc`,`malloc`

쌇farrealloc

#함수 FARREALLOC

기 능 far heap 에 할당된 블록의 크기를 조정한다.

구 문 void far *farrealloc(void far *oldblock, unsigned long nbytes);

프로토타입 `alloc.h`

기능 설명 할당된 블록의 크기를 nbytes 로 재조정하며 필요한 경우에 블록의 내용을 새로운 내용으로 대체한다. far heap 으로 부터 블록을 할당하면,

* 모든 사용 가능한 RAM 을 할당할 수 있다.
* 64K 이상의 메모리 블록을 할당할 수 있다.
* Far 포인터로 블록을 억세스 한다.

tiny 모델에서는 .COM 파일로 변환되지 않으면 사용될 수 없다.

리 턴 값 다시 할당된 블록의 어드레스를 리턴하는 데 최초 블록의 어드레스와 다를 수도 있다.

이 식 성 DOC 에 한한다.

참 조 `farmalloc`,`realloc`

쌇free

#함수 FREE

기 능 할당된 블록을 해제한다.

구 문 void free(void *block);

프로토타입 `stdlib.h`,`alloc.h`

기능 설명 calloc,malloc,realloc 등의 사전호출에 의해서 할당되어 있는 메모리 블록을 해제시킨다.

리 턴 값 없음

이 식 성 UNIX 시스템에서 사용할 수 있으며,ANSI-C 와

도 호환 된다.

참 조 `calloc`,`freemem`,`malloc`,`realloc`, `strdup`

쌇malloc

#함수 MALLOC

기 능 메인 메모리를 할당한다.

구 문 void *malloc(size_t size);

프로토타입 `stdlib.h`,`alloc.h`

기능 설명 C 메모리의 heap 으로 부터 size 만큼의 블록을 할당한다. 이 함수는 프로그램이 메모리를 필요로 하는 양만큼 명시적으로 할당할 수 있도록 해준다.

heap 은 가변 크기 블록의 동적할당에 사용된다. 트리나 리스트와 같은 데이타구조는 거의가 heap 메모리 할당을 사용하게 된다.

데이타 세그먼트 맨 끝과 프로그램의 스택의 256바이트의 여분을 제외 하고는 유용하게 쓰일 수 있다. 이 여분은 DOS 에 필요한 소량의 영역이외에 어플리케이션 프로그램을 위해 스택을 증대 시킬 수 있도록 하는 영역을 허용한다.

large 데이타 모델 에서는 프로그램이 스택을 지나 물리적 메모리 끝까지 모든 공간이 heap 을 위해 사용할 수 있는 공간이 된다.

리 턴 값 에러가 없으면 malloc 은 새로이 할당된 메모리 블록의 포인터를 리턴한다. 새로운 블록을 위한 충분한 공간이 없는 경우에는 NULL을 리턴한다. 이 블록의 내용은 변경되지 않는다. 인수 size가 0인 경우에도 NULL 이 리턴된다.

이 식 성 UNIX 시스템에서 사용될수 있으며, ANSI-C 와도 호환 된다.

참 조 `allocmem`,`calloc`,`coreleft`,`farcalloc`, `farmalloc`,`free`,`realloc`

쌇realloc

#함수 REALLOC

기　　능　메인 메모리를 다시 할당한다.
구　　문　void *realloc(void *block, size_t size);
프로토타입　`stdlib.h`,`alloc.h`
기능 설명　이미 할당된 블록의 크기를 size 에 맞게 조
정 한다. block 은 malloc,calloc 또는 real
loc 에 의해서 할당된 블록을 가르킨다. 만일
block의 값이 NULL 이면 realloc 의 기능은 m
alloc 과 같다. realloc은 할당된 블록의 크
기를 size 에 맞추고 필요한 경우에 이전 블
록의 내용을 새로운 위치로 복사한다.
리 턴 값　재할당된 블록의 어드레스를 리턴한다. 이 어
드레스는 처음의 블록의 값과 다를수고 있다.
블록의 재할당에 실패하거나 size 의 값이 0
이면 NULL을 리턴한다.
이 식 성　UNIX 시스템에서 사용할 수 있으며 ANSI-C 와
도 호환된다.
참　　조　`calloc`,`farrealloc`,`free`,`malloc`

쌿sbrk
#함수 SBRK

기　　능　데이타 세그먼트 할당을 변경한다.
구　　문　void *sbrk(int incr);
프로토타입　`alloc.h`
기능 설명　break 값에 incr 바이트를 더하고 할당된 공
간을 적절하게 변경한다. incr 은 음수일 수
도 있으며 이 때는 할당된 공간의 양이 감소
하게 된다.
만약 sbrk에 요구된 공간이 할당 가능한 양을
넘어 서면 sbrk는 할당된 공간에 아무런 변경
도 하지 못할 수도 있다.
리 턴 값　에러가 없으면 sbrk는 기존이 break 값을 리
턴 하고 에러시는 -1을 리턴한다.
`errno`은 다음값으로 설정된다.

ENOMEM　메모리(core) 가 충분치 않다.
이 식 성　UNIX 시스템에서 사용할 수 있다.
참　　조　`brk`

쌿NULL

#의사변수 NULL

의　　미　보통 포인터 변수는 어떤 정보가 수록 되어있는 어드레스를 가리킨다. NULL 은 포인터 변수가 아무 것도 가리키고 있지 않음을 나타낸다.

정　　의　`alloc.h`,`mem.h`,`stddef.h`,
`stdio.h`,`stdlib.h`

쌇size_t
#파라미타 size_t

메모리 블록에 관계된 함수에서 메모리 블록의 크기나,갯수를 나타내는 파라미터로 사용된다.

정　　의　`alloc.h`,`mem.h`,`stddef.h`,
`stdio.h`,`stdlib.h`,`string.h`

쌇assert.h
#헤더파일 ASSERT.H

함수

`assert`

쌇assert
#함수 ASSERT

기　　능　조건을 검사한 후 중단 시킨다.
구　　문　#include<assert.h>
#include<stdio.h>
void assert(int test);
프로토타입　`assert.h`
기능 설명　if 명령을 전개하는 매크로 이다. test가 0이면 assert는 stderr에 메시지를 프린트 하고 프로그램을 중단 시킨다.(abort 호출을 통해) assert 는 다음 메시지를 프린트 한다.

Assertion failed: <text>, file <filename>
, line <linenum>

메시지에 리스트된 filename과 linenum은 ass
ert 매크로가 나타나는 곳의 소스파일과 행번
호 이다. #include<assert.h> 지시어 앞에 소
스 코드에 #define NDEBUG 지시어('디버깅 하
지 않는다.)를 위치 시키면 assert 명령이 co
mment 문으로 취급된다.

리 턴 값 없음
이 식 성 시스템 III와 V를 포함하는 일부 UNIX 시스템
상에서 사용할 수 있다. ANSI-C 와도 호환된
다.
참 조 `abort`

쌿bios.h
#헤더파일 BIOS.H

함수

`bioscom` `biosmemory`
`biosdisk` `biosprint`
`biosequip` `biostime`
`bioskey`

쌿bioscom
#함수 BIOSCOM

기 능 직렬식 입출력(I/O)을 수행한다.
구 문 int bioscom(int cmd,char abyte,int port);
프로토타입 `bios.h`
기능 설명 bioscom은 port에서 주어진 입출력 포트를 통
해 다양한 RS-232 통신을 수행한다.
port값 0은 COM1에 대응하고 port값 1은 COM2
... 와 같은 형태로 대응된다. cmd 값은 다음
중 하나가 된다.

0 통신 파리미터를 abyte 값으로 변경한다.
1 통신 라인을 통해 문자 abyte 를 보낸다.
2 통신 라인으로 부터 문자를 받는다.
3 통신 포트의 현재 상태를 리턴한다.

abyte는 다음 비트의 조합이다. (하나의 값은
그룹의 각각으로 부터 선택된 것이다.)

0x02 7 data bits
0x03 8 data bits

0x00 1 stop bit
0x04 2 stop bits

0x00 No parity
0x08 Odd parity
0x18 Even parity

0x00 110 baud
0x20 150 baud
0x40 300 baud
0x60 600 baud
0x80 1200 baud
0xA0 2400 baud
0xC0 4800 baud
0xE0 9600 baud

bioscom 은 BIOS 0x14 인터럽트를 사용한다.

리 턴 값 모든 cmd 값에 대해 bioscom은 16비트 정수를
리턴하는데 상위 8비트는 상태비트 이고 하위
8비트는 cmd값에 따라 다양하게 변화된다.

이 식 성 IBM-PC와 그 호환 계열에서 작동된다.

쌇biosdisk
#함수 BIOSDISK

기 능 BIOS 디스크 서비스

구 문 int biosdisk(int cmd, int drive, int head
int sector, int nsetcs, void *buffer);

프로토타입 `bios.h`

기능 설명 BIOS에 대해 직접 디스크를 조작하기 위하여
인터럽트 0x13 을 사용한다. drive는 사용될
드라이브를 지정하는 수 이다.

플로피 드라이브 에서는 0이 첫번째 1이 두번
째... 와 같은 식으로 설정되고 하드 디스크

에서는 0x80이 첫번째,0x81이 두번째.. 와 같
이 설정된다.

cmd는 수행해야 할 동작을 지시한다. 이 cmd
값에따라 다른 파라미터가 필요할 수도 있다.
다음은 cmd 값에 따른 biosdisk의 동작이다.

0 드라이브 콘트롤러 하드를 리셋시킨다.
1 마지막 디스크 동작상태를 리턴한다.
2 메모리로 한개 이상의 섹터를 읽어온다.
3 한개 이산의 섹터를 디스크에 쓴다.
4 한개 이산의 섹터를 검사한다.
5 트랙을 포맷한다.

리 턴 값 작업이 제대로 수행되면 0을 리턴한다.
이 식 성 IBM-PC호환 계열에서 작동한다.
참 조 `absread`,`abswrite`

쌿biosprint
#함수 BIOSPRINT

기 능 BIOS를 직접 사용하여 프린터에 입출력 한다.
구 문 int biosprint(int cmd, int abyte,
int port);
프로토타입 `bios.h`
기능 설명 BIOS 인터럽트 0x17을 사용하여,port에 의해
정의된 프린트 상에 여러가지 프린트 기능을
수행한다. port값 0은 LPT1에 대응되고 port
값 1은 LPT2에 대응된다. cmd값은 다음중 하
나가 된다.

0 abyte 문자를 프린트 한다.
1 프린트 포트를 초기화 한다.
2 프린터 상태를 읽는다.

abyte 값은 0에서 255까지 될 수 있다.

리 턴 값 리턴되는 값은 다음의 비트값을 논리합 으로
조합 함으로써 현재의 프린터 상태가 된다.

Bit 0 0x01 디바이스 종료

Bit 3 0x08 입출력(I/O) 에러
Bit 4 0x10 선택됨
Bit 5 0x20 종이 부족
Bit 6 0x40 인식
Bit 7 0x80 작동 가능

이 식 성 IBM-PC와 호환 계열에서만 작동된다.

쌿biosequip
#함수 BIOSEQUIP

기 능 장비를 검사한다.
구 문 int biosequip(void);
프로토타입 `bios.h`
기능 설명 시스템에 연결되어 있는 장비를 나타내는 정수를 리턴한다. 이를 위해 BIOS 인터럽트 0x11을 사용한다
이 식 성 IBM-PC 와 호환 계열에서만 작동한다.

쌿bioskey
#함수 bioskey

기 능 키보드 인터페이스로 바이오스를 직접 사용한다.
구 문 int bioskey(int cmd);
프로토타입 `bios.h`
기능 설명 BIOS 인터럽트 0x16을 사용하여 다양한 키보드 조작을 수행한다. 파라미터 cmd는 조작을 결정한다.
리 턴 값 bioskey에 의해 리턴되는 값은 수행 작업 (cmd값에 의해서 결정됨)에 따른다.

cmd 값에 의해서 행해지는 작업	
0	하위 8비트가 0이 아닐경우 bioske는 큐에서 대기하고 있는 다음 키스트로크나, 키보드에서 타이핑 된후 키를 위한 ASCII문자 값을 리턴한다. 하위 8비트가 0일 경우 상위 8비트는 확장 키보드 코드가 된다.
1	키 스트로크가 읽혀 질 수 있는지 검사한다. 0이 리턴되면 키 보드가 눌려지지 않았음을 의미한다. 그렇지 않으면 다음의

키 값이 리턴된다.
2 현재의 Shift 키 상태를 리턴한다.

이 식 성 IBM-PC 호환 계열에서만 작동한다.

쌇biosmemory
#함수 BIOSMEMORY

기 능 RAM의 크기를 리턴한다.
구 문 int biosmemory(void);
프로토타입 `bios.h`
기능 설명 BIOS 인터럽트 0x12를 사용하여 RAM의 크기를 리턴한다. 이 값은 디스플레이 어댑터와 확장 메모리는 포함하지 않는다.
리 턴 값 1K 블럭 내에서 RAM의 크기를 리턴한다.
이 식 성 IBM-PC 호환 계열에서만 작동한다.

쌇biostime
#함수 BIOSTIME

기 능 BIOS 타이머를 읽거나 변경한다.
구 문 long biostime(int cmd,long newtime);
프로토타입 `bios.h`
기능 설명 biostime은 BIOS 타이머를 읽거나 설정한다. 이 값은 자정이 지난 후 부터 초당 18.2 번의 비율로 tick을 계산하는 타이머 이다.

biostime은 BIOS인터럽트 0x1A를 사용한다.
cmd가 0이면 현재 타이머의 값을 리턴한다.
cmd가 1이면 현재 타이머의 값을 newtime으로 설정한다.
리 턴 값 cmd가 0이면 현재의 타이머 값을 리턴한다.
이 식 성 IBM-PC 호환 계열에서만 작동한다.

쌇conio.h
#헤더파일 CONIO.H

함수

`cgets` `clreol` `clrscr`
`cprintf` `cputs` `cscanf`

`delline` `getch` `getche`
`getpass` `gettext` `gettextinfo`
`gotoxy` `highvideo` `insline`
`kbhit` `lowvideo` `movetext`
`normvideo` `putch` `puttext`
`textattr` `textbackground` `textcolor`
`textmode` `ungetch` `wherex`
`wherey` `window`

상수, 데이타형, 파라미터

`BLINK` `COLORS` `directvideo`
`text_info` `text_modes`

쌿cgets
#함수 CGETS

기 능 콘솔로부터 문자열을 일거 들인다.
구 문 char *cgets(char *str);
프로토타입 `conio.h`
기능 설명 콘솔(키보드)로 부터 문자열을 읽어들이고 str 이 가르키는 장소에 문자열을 저장하는 것 이다. 또한 cgets는 최대로 읽어들일 수 있는 문자수, 혹은 CR/LF라고 하는 조합을 만날 때까지 문자를 읽어 들인다. cgets가 CR/LF 조합을 읽어 들이면 그것은 문자열을 조합하기 전에 "\0"(null-terminator)으로 조합을 대체한다.

cgets가 호출되기 전에, str[0]은 읽어 들일문자의 최대 길이로 설정 되어 있어야 한다. cgets의 처리가 종료된 후에는 str[1]에 실제로 읽어들인 문자의 수가 리턴된다. 따라서 실제로 읽어 들인 문자는 str[2] 에서 시작해서 null종료까지 이다. 따라서 cgets를 호출하기 위해서는 최소 str[0]+2의 길이가 필요 하게 된다.

리 턴 값 str[2]에 대한 포인터를 리턴하며 특별한 에러 리턴은 없다.
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환계열 에서 사용된다.

참　　조 `fgets`,`getch`,`getche`,`gets`

쌇clreol
#함수 CLREOL

기　　능 텍스트 윈도 내에서 행 끝까지 소거한다.
구　　문 void clreol(void);
프로토타입 `conio.h`
기능 설명 clreol은 현재의 텍스트 윈도내에서 커서의 이동 없이 커서 위치에서 핸 끝까지의 문자를 소거 한다.
리 턴 값 없음
이 식 성 IBM-PC 호환 계열에서만 작동한다.
참　　조 `clrscr`,`delline`,`window`

쌇clrscr
#함수 CLRSCR

기　　능 텍스트 모드상의 윈도를 클리어 한다.
구　　문 void slrscr(void);
프로토타입 `conio.h`
기능 설명 현재의 키보드 윈도를 클리어 하고 커서를 좌상귀 위치(1,1)로 이동 시킨다.
리 턴 값 없음
이 식 성 IBM-PC 호환 계열에서만 작동한다.
참　　조 `clreol`,`delline`,`window`

쌇cprintf
#함수 CPRINTF

기　　능 화면에 서식화(formatted) 출력을 내보낸다.
구　　문 int cprintf(
　　const char*format [,argument,...]);
프로토타입 `conio.h`
기능 설명 일련의 인수를 받아 들여서 format이 가르키는 포맷 문자열에 있는 포맷 지정을 각각에 지정하고 서식화된 데이타를 화면상에 현재 텍스트 윈도우에 출력한다. 따라서 인수 만큼 포맷 지정의 수가 있어야 한다.

포맷 지정에 대한 자세한 정보는`printf`함수

를 참고 하기 바란다.

여기서 cprintf는 printf와 똑같이 작동하지만 그 출력이 표준 출력이 아닌 콘솔에 대해서 수행 된다는 점이 다르다. 따라서 printf문의 다른 행 문자(\n) 은 (\r\n)으로 바뀌어 씌어 져야 한다.

리 턴 값 출력 문자의 수를 리턴한다.
이 식 성 IBM-PC 호환 계열에서만 작동한다.
참 조 `directvideo`,`fprintf`,`printf`,`putch`, `sprintf`,`vprintf`

쌿cputs
함수 CPUTS

기 능 화면에 한개의 문자열을 출력한다.
구 문 int cputs(const char *str);
프로토타입 `conio.h`
기능 설명 cputs는 널 문자가 있는 문자열 str을 현재의 텍스트 윈도에 출력한다. 이것은 새로운 행의 문자를 추가 하지는 않는다. 문자열은 direct video의 값에 따라서 BIOS 호출을 통해 메모리에 직접 출력된다.

메모리에 직접 문자를 넣으므로 '\n'이나'\r' 과 같은 제어 문자열은 사용할 수 없다.

리 턴 값 출력된 마지막 문자를 리턴한다.
이 식 성 IBM-PC 호환 계열에서만 작동한다.
참 조 `directvideo`,`putch`,`puts`

쌿cscanf
함수 CSCANF

기 능 콘솔로부터 스캔과 포맷에 의한 입력을 한다.
구 문 int cscanf(char *format[, address,...]);
프로토타입 `conio.h`
기능 설명 한번에 한 문자씩 콘솔로 부터 직접 읽어들이면서 일련의 입력 필드를 스캔(scan)한다.
이때 각 필드는 format 이 지정하는 format문자열에서 cscanf에 전달된 포맷 지정에 따라 나타난다.

최종적으로 cscanf는 format다음에 오는 인수가 지정한 어드레스에 포맷된 입력을 저장하고 그것을 화면에 에코 (display) 한다. 포맷 지정에 관한 사항은`scanf`를 참조한다.

여러가지 이유로 scanf는 특정한 필드에 대해서 필드 끝(end-of-field: 공백) 문자에 이르기 전에 스캐닝을 중지 하거나 완전히 종료하기도 한다. 이에 대해서는`scanf`를 참고하기 바란다.

리 턴 값 성공적으로 스캔(scan) 되면 변환되어 저장된 입력 필드의 수를 리턴한다. 이 리턴 값은 저장 되지 않은 채 스캔된 필드는 포함하지 않는다.

어떤 필드도 지정되지 않은 경우 리턴값은 0이다. cscanf가 파일종료를 읽어들이는 경우 리턴 값은`EOF`가 된다.

이 식 성 UNIX 시스템 상에서 사용할 수 있다. 이것은 Kernighan과 Ritchie에서 정의 된다.

참 조 `fscanf`,`getche`,`scanf`,`sscanf`

쌇delline

#함수 DELLINE

기 능 텍스트 윈도에서 행을 삭제한다.

구 문 void delline(void);

프로토타입 `conio.h`

기능 설명 커서가 있는 행을 삭제한 다음 그 밑에 있는 행을 위로 끌어 올린다. deline은 현재 진행중인 텍스트 윈도에서 작동된다.

리 턴 값 없음

이 식 성 IBM-PC와 호환기종 에서 만 작동한다.

참 조 `clreol`,`clrscr`,`insline`,`window`

쌇getch

#함수 GETCH

기 능 키보드로 부터 문자를 읽어 들이고 화면 에는 에코 하지 않는다.

구　　문　int getch(void);
프로토타입　`conio.h`
기능 설명　키보드로 부터 문자를 읽어 들이지만 화면 에는 에코 하지 않는다. getch는 stdin을 사용 한다.
리 턴 값　키보드로 부터 읽는 문자를 리턴한다.
이 식 성　DOS에 한한다.
참　　조　`cgets`,`fgets`,`getc`,`getchar`,`getche`, `getpass`,`kbhit`,`putch`,`ungetch`

쌇getche
#함수 getche

기　　능　콘솔로 부터 문자를 읽어 들이고 화면에 에코 한다.
구　　문　int getche(void);
프로토타입　`conio.h`
기능 설명　콘솔로부터 단일 문자를 읽어 들인 다음, 현재의 텍스트 윈도에 에코한다. 직접 비디오에 엑세스 하거나(`directvideo`참조) BIOS를 이용 한다.
리 턴 값　키보드로 부터 읽어 들인 문자를 리턴한다.
이 식 성　DOS에 한한다.
참　　조　`gets`,`cscanf`,`fgetc`,`getc`,`getch`, `getchar`,`kbhit`,`putch`,`ungetch`

쌇getpass
#함수 GETPASS

기　　능　패스워드를 읽는다.
구　　문　char *getpass(const char *prompt);
프로토타입　`conio.h`
기능 설명　시스템 콘솔로 부터 패스워드를 읽어 들인다. 이 때 null종료 문자열 prompt를 표시하고 에코는 금지 시킨다.

앞에서 8자 까지 유효하고 null 종료 문자열의 포인터가 리턴된다. 이때 null종료기는 포함 되지 않는다.
리 턴 값　리턴값은 정적 문자열의 포인터가 되고 이것은 호출시 마다 변경된다.

이 식 성 UNIX 시스템 상에서 사용할 수 있다.
참 조 `getch`

쌇gettext
#함수 GETTEXT

기 능 텍스트 모드 화면으로 부터 텍스트를 메모리
로 복사한다.
구 문 int gettext(int left, int top, int right,
int bottom, void *destin);
프로토타입 `conio.h`
기능 설명 (left,top)과 (right,bottom)은 대각으로 으
로 하는 장방향 영역의 텍스트의 내용을 des
tin이 지정된 메모리에 복사한다. 모든 좌표
는 윈도의 상대좌표가 아닌 절대 좌표가 도
는데 좌상귀는 (1,1) 이 된다.

gettext는 장방향의 내용을 좌에서 우로, 위
에서 아래로 순서적으로 메모링 읽어온다.
화면상의 각 위치는 2바이트 메모리를 차지
한다. 여기서 첫번째 바이트는 셀에서의 문자
이고 두번째는 셀의 비디오 어트리뷰트가 된
다. w 행과 h 열인 장방향이 차지하는 공간의
바이트 수는 아래와 같이 계산하면 된다.

bytes = (h열) X (w행) X 2
리 턴 값 에러가 없으면 1을 리턴하고 에러가 나면(가
령 현재 화면모드 범위 밖의 값을 주면) 0을
리턴 한다.
이 식 성 IBM-PC와 BIOS호환 시스템 에서만 작동된다.
참 조 `movetext`,`puttext`

예 제 char buf[20*10*2]
/* 화면을 저장한다. */
gettext(1,1,20,10,buf);
/* 화면을 복구한다. */
puttext(1,1,buf);

쌇gettextinfo
#함수 GETTEXTINFO

기 능 텍스크모드 비디오 정보를 구한다.
구 문 #include<conio.h>
void gettextinfo(struct text_info *r);
프로토타입 `conio.h`
기능 설명 r이 가르키는 text_info 구조체에 현재의 비디오 정보를 채워 넣는다. text_info 구조체는 conio.h 에서 다음과 같이 정의 된다.

```
struct text_info {
        unsigned char winleft;
        unsigned char wintop;
        unsigned char winright;
        unsigned char winbottom;
        unsigned char attribute;
        unsigned char normattr;
        unsigned char currmode;
        unsigned char screenheight;
        unsigned char screenwidth;
        unsigned char curx;
        unsigned char cury;
};
```

리 턴 값 화면상에 리턴 하지는 않는다. 결과는 r이 가리키는 구조체에 리턴된다.
이 식 성 IBM-PC와 호환기종에서 작동된다.
참 조 `textattr`,`textbackground`,`textcolor`, `textmode`,`wherex`,`wherey`,`window`

예 제

```
#include <conio.h>
struct text_info tinfo;
main()
{
  /* 텍스트 정보를 받아들인다. */
  gettextinfo(&tinfo);
  /* 텍스트 정보의 화면모드로 복귀한다. */
  textmode(tinfo,currmode);
}
```

쌿gotoxy
#함수 GOTOXY

기 능 텍스트 윈도에서 커서의 위치를 설정한다.

구　　문　void gotoxy(int x,int y);
프로토타입　`conio.h`
기능 설명　텍스트 윈도에서 커서를 주어진 위치로 이동 시킨다. 지정된 좌표가 윈도를 벋어 나는등 잘 못된 좌표가 주어지면 gotoxy 호출은 무시 된다. 따라서 커서도 이동하지 않게 된다. 예를 들어 커서의 우하귀가 (80,25)인 경우에 gotoxy(90,25)를 호출하는 경우가 이에 해당 된다.
리 턴 값　없음
이 식 성　IBM-PC 와 호환기종 에서만 작동한다. 비슷한 함수가 터보 파스칼에도 있다.
참　　조　`wherex`,`wherey`,`window`
예　　제　/* 커서를 10열 20행으로 이동 시킨다. */
gotoxy(10,20);

쌇highvideo
#함수 HIGHVIDEO

기　　능　고휘도(high-insity) 문자를 선택한다.
구　　문　void highvideo(void);
프로토타입　`conio.h`
기능 설명　선택된 전경 색상을 고휘도 비트로 설정함으로써 고휘도 문자를 선택할 수 있다. 이 함수는 이미 화면상에 있는 문자에는 영향을 미치지 않고 highvideo 가 호출된후에 화면상에 텍스트모드 출력을 하는`cprintf`와 같은 함수에 영향을 미치게 된다.
리 턴 값　없음
이 식 성　IBM-PC와 호환계열 에서만 작동한다. 대응 되는 함수가 터보 파스칼에도 있다.
참　　조　`lowvideo`,`normvideo`,`textattr`, `textcolor`

쌇insline
#함수 INSLINE

기　　능　텍스트 윈도에 공백행을 삽입한다.
구　　문　void insline(void);
프로토타입　`conio.h`
기능 설명　텍스트 윈도 내의 커서 위치에서 현재의 텍스

트 배경색 으로 공백행을 1행 삽입한다. 그 자리에 원래있던 핸은 새로 삽입되는 행의 아래로 한 줄씩 밀려난다. insline은 텍스트 모드 에서 사용된다.

리 턴 값 없음

이 식 성 IBM-PC와 호환계열에서 작동한다. 터보 파스칼에도 대응되는 함수가 있다.

참 조 `delline`,`window`

쌿kbhit

#함수 KBHIT

기 능 키보드로 부터의 입력 여부를 검사한다.

구 문 int kbhit(void);

프로토타입 `conio.h`

기능 설명 키보드로부터의 입력 여부를 검사한다. 사용된 키보드는 getch나 getche로 검출될 수 있다.

리 턴 값 키보드의 입력이 있었으면 1을 리턴하고 입력이 없었으면 0을 리턴한다.

참 조 `getch`,`getche`

쌿lowvideo

#함수 LOWVIDEO

기 능 저휘도(low-intensity)문자를 선택한다.

구 문 void lowvideo(void);

프로토타입 `conio.h`

기능 설명 현재 선택된 텍스트 모드 전경 색상을 저휘도 문자로 설정한다. 이 함수는 이미 화면상에 있는 문자에 대해서는 영향을 미치지 않으며 cprintf와 같은 화면상에 문자를 나타내는 함수에 영향을 미친다.

리 턴 값 없음

이 식 성 IBM-PC와 호환기종 에서만 작동한다. 동일한 기능을 하는 함수가 터보 파스칼에도 있다.

참 조 `highvideo`,`normvideo`,`textattr`, `textcolor`

쌿movetext

#함수 MOVETEXT

기 능 직사각형 안의 텍스트를 다른곳으로 복사한다.

구 문 int movetext(int left, int top, int right,
int bottom,int destleft,
int desttop);

프로토타입 `conio.h`

기능 설명 left, top, right, bottop 에 의해서 정의된 직사각형 안의 텍스트를 이동 복사 시킨다. 이 새로운 영역은 (destleft , desttop) 을 좌상귀로 하는 직사각형 이다. 모든 좌표는 화면상의 절대 좌표로 부여된다. 복사영역이 중복 되어도 정확하게 복사된다.

리 턴 값 에러가 없으면 0 이외의 값을 리턴하지만, 그렇지 않으면(가령 현재 화면 모드의 범위 밖에의 좌표를 지정하는 경우) 0을 리턴한다.

이 식 성 IBM-PC와 BIOS 호환 시스템 에서만 사용할 수 있다.

참 조 `gettext`,`puttext`

예 제
```
/* 화면좌표 (10,20,20,30) 의 영역을
   (15,25)로 이동 시킨다. */
movetext(10,20,20,30,15,25);
```

쌇normvideo

#함수 NORMVIDEO

기 능 문자를 정상적인 명암도로 선택한다.

구 문 void normvideo(void);

프로토타입 `conio.h`

기능 설명 텍스트 속성 (전경색과 배경색)을 프로그램이 시작할 당시 가졌던 값을 리턴함으로써 정상적인 문자를 선택한다. 이 함수는 현재 화면상에 있는 문자에는 영향을 미치지 않고 함수 normvideo가 수행된 후에 직접 콘솔입력을 수행 하는 함수(`cprintf`같은)에 의해서 디스플레이 되는 문자에만 영향을 미치게 된다.

리 턴 값 없음

이 식 성 IBM-PC와 호환기종 에서만 작동된다. 터보 파스칼에도 대응되는 함수가 있다.

참　　조　`highvideo`,`lowvideo`,`textattr`,
　　`textcolor`

쌇putch
#함수 PUTCH

기　　능　문자를 화면에 출력한다.
구　　문　int putch(int c);
프로토타입　`conio.h`
기능 설명　문자 c를 현재 텍스트 윈도에 출력한다. 이것은 콘솔에 직접 비디오 출력을 수행하는 텍스트 모드 함수이다 putch는 (/n) 을 (/n/r) 로 해석하지 않는다.
리 턴 값　putch는 출력되는 문자 c를 리턴한다. 만약 에러가 발생되면`EOF`를 리턴한다.
이 식 성　IBM-PC와 호환 기종에서만 작동한다.
참　　조　`cprintf`,`cputs`,`getch`,`getche`,`putc`,
　　`putchar

쌇puttext
#함수 PUTTEXT

기　　능　메모리로부터 텍스트 내용을 텍스트 모드 화면에 출력한다.
구　　문　int puttext(int left, int top, int right,
　　int bottom,void *source);
프로토타입　`conio.h`
기능 설명　source 가 지정하는 메모리 영역의 내용을 left,top과 right,bottom을 대각으로 하는 장방향의 영역에 출력한다. 모든 좌표는 윈도의 상대좌표가 아니고 절대 화면 좌표 이다. 좌상귀의 좌표는 (1,1)이다. puttext는 메모리 영역의 내용을 좌우상하의 순서로 정의된 사각형 안에 순서대로 정리시킨다.
리 턴 값　에러가 없으면 0이 아닌 값을 리턴하고 이외에는(예를 들어 사용자가 현재 화면 모드 범위 밖의 좌표를 주는것 같은) 0을 리턴한다.
이 식 성　IBM-PC와 BIOS 호환 시스템상에서만 작동한다.
참　　조　`gettext`,`movetext`,`window`

쌇textattr
#함수 TEXTATTR

기　　능　텍스트의 속성을 결정한다.
구　　문　void textattr(int newattr);
프로토타입　`conio.h`
기능 설명　한 번 호출으로 문자의 배경색과 전경색을 설정한다. (사용자는 textcolor와 textbackground 를 사용하여 각각 전경색과 배경색을 선택할 수 있다.) 이 함수는 현재 화면상에 있는 문자에는 영향을 미치지 않고 함수 textattr 이 수행된 후에 직접 콘솔입력을 수행 하는 함수(`cprintf`같은)에 의해서 디스플레이 되는 문자에만 영향을 미치게 된다.

색상정보는 아래와 같은 형태가 되어야 한다.

```
  7   6   5   4   3   2   1   0
----------------|----------------
| B | b | b | b | f | f | f | f |
----------------|----------------
```

8비트 newattr 파라미터
ffff 4비트의 전경색(0에서 15)
bbb 3비트의 배경색(0에서 7)
B 반짝거림 비트

만약 깜박거림 비트가 1이면 해당 문자를 깜박 거린다. 이것은 해당 속성에 BLINK를 더함으로써 얻어진다.

리 턴 값　없음
이 식 성　IBM-PC와 호환기종 에서만 사용된다.
참　　조　`gettextinfo`,`highvideo`,`lowvideo`, `normvideo`,`textbackground`,`textcolor`
예　　제

```
/* 텍스트의 색을 파란색 배경에 노란색 깜빡
   거리는 문자로 선택한다. */
```

쌇textbackground
#함수 TEXTBACKGROUND

기　　능　새로운 텍스트 배경색을 선택한다.
구　　문　void textbackground(int newattr);

프로토타입 `conio.h`
기능 설명 텍스트의 배경색을 새로 설정한다. 이후에 텍스트 모드에서 비디오 직접 출력을 행하는 함수에 의해 기록되는 문자의 배경색이 0-7까지의 정수는 newattr에 의해서 결정된다.

사용자는 conio.h에 정의된 기호 상수명을 사용하여 색상을 표시할 수 있다. 만약 사용자가 이들을 사용하려면 conio.h를 포함하여야 한다. 이 함수는 현재 화면상에 있는 문자에는 영향을 미치지 않고 함수 textattr이 수행된 후에 직접 콘솔입력을 수행 하는 함수 (`cprintf`같은)에 의해서 디스플레이 되는 문자에만 영향을 미치게 된다.

아래의 표는 가능한 색상과 수치값이다.

기호 상수명	수치
BLACK	0
BLUE	1
GREEN	2
CYAN	3
RED	4
MAGENTA	5
BROWN	6
LIGHTGRAY	7

리 턴 값 없음
이 식 성 IBM-PC와 호환기종 에서만 동작되며 터보 파스칼 에도 대응되는 함수가 있다.
참 조 `gettextinfo`,`textattr`,`textcolor`

쌇textcolor
#함수 TEXTCOLOR

기 능 텍스트 모드에서 문자의 색을 새로 선택한다.
구 문 void textcolor(int newattr);
프로토타입 `conio.h`
기능 설명 전경문자의 색을 설정한다. 이후에 콘솔 출력 함수에 의해서 화면에 써지는 모든 문자의 색은 newattr이 된다. 사용자는 색상을 conio.h

에 정의된 상수명을 이용하여 나타낼 수 있다 만약 이러한 상수명을 사용하려면 사용자는 conio.h 파일을 포함하여야 한다. 이 함수는 현재 화면상에 있는 문자에는 영향을 미치지 않고 함수 texcolor가 수행된 후에 직접 콘솔 출력을 수행 하는 함수 (`cprintf`같은)에 의해서 디스플레이 되는 문자에만 영향을 미치게 된다.

리 턴 값 없음

이 식 성 IBM-PC와 호환기종에서만 작동한다. 동일한 함수가 터보 파스칼에도 있다.

참 조 `gettextingo`,`highvideo`,`lowvideo`,`normvideo`,`textattr`,`textbackground`

쌿textmode

#함수 TEXTMODE

기 능 화면을 텍스트 모드로 바꾼다.

구 문 void textmode(int nremode);

프로토타입 `conio.h`

기능 설명 지정된 텍스트 모드를 선택한다. 사용자는 conio.h에 정의된 text_mode 기호 상수명을 이용 하여 인수 newmode를 지정할 수 있다. 만일 사용자가 이러한 상수명을 사용하려고 한다면 conio.h를 포함하여야 한다. 다음표에 text_modes에 속한 명칭과 그 값이 있다.

기호상수명	수치	텍스트모드
LASTMODE	-1	이전의 텍스트 모드
BW40	0	흑백 40열
C40	1	칼라 40열
BW80	2	흑백 80열
C80	3	칼라 80열
MONO	7	단색(모노크롬) 80열

리 턴 값 없음

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC와 호환기종 에서 사용할 수 있다.

참 조 `gettextinfo`,`window`

쌇ungetch
#함수 UNGETCH

기　　능　한 문자를 키보드 버퍼로 되돌린다.
구　　문　int ungetch(int ch);
프로토타입　`conio.h`
기능 설명　문자 ch가 다음에 읽려지도록 콘솔에 되돌린다. 만일 ungetch가 연속으로 불려지면 에러가 발생한다.
리 턴 값　에러가 없으면 문자 ch를 리턴하며 에러가 발생 하면`EOF`를 리턴한다.
이 식 성　UNIX 시스템 상에서 사용할 수 있다.
참　　조　`getch`,`getche`

쌇wherex
#함수 WHEREX

기　　능　윈도 내에서 커서 수평 위치를 구한다.
구　　문　int wherex(void);
프로토타입　`conio.h`
기능 설명　현재의 텍스트 윈도내에서 커서의 좌표 x 를 리턴한다.
리 턴 값　1에서 80까지의 정수를 리턴한다.
이 식 성　IBM-PC와 호환기종에서만 작동한다. 동일한 함수가 터보 파스칼에도 있다.
참　　조　`gettextinfo`,`gotoxy`,`wherey`

쌇wherey
#함수 WHEREY

기　　능　윈도 내에서 커서 수직 위치를 구한다.
구　　문　int wherey(void);
프로토타입　`conio.h`
기능 설명　현재의 텍스트 윈도내에서 커서의 좌표 y 를 리턴한다.
리 턴 값　1에서 25까지의 정수를 리턴한다.
이 식 성　IBM-PC와 호환기종에서만 작동한다. 동일한 함수가 터보 파스칼에도 있다.
참　　조　`gettextinfo`,`gotoxy`,`wherex`

쌇window
#함수 WINDOW

기　　능　텍스트 모드에서의 윈도를 정의한다.
구　　문　void window(int left, int top, int right
　　int bottom);
프로토타입　`conio.h`
기능 설명　화면상에서 텍스트 윈도를 정한다. 좌표의 값이 적절치 못할 경우에 window 호출은 무시된다.
　　left,top 은 화면의 좌상귀를 나타낸다.
　　right,bottom 은 화면의 우하귀를 나타낸다.
　　윈도의 최소크기는 1행 1칸이다. 윈도를 지정하지 않으면 디폴트로 전화면에 윈도로 설정된다.
　　이들 좌표는 아래와 같다.
　　80 컬럼 모드 : 1,1,80,25
　　40 컬럼 모드 : 1,1,40,25
리 턴 값　없음
이 식 성　IBM-PC와 호환기종에서만 작동한다. 동일한 함수가 터보 파스칼에도 있다.
참　　조　`clreol`,`clrscr`,`delline`,`gettextinfo`,
　　`gotoxy`,`insline`,`puttext`,`textmode`

쌇BLINK
#상수 BLINK

주로 텍스트계 함수에서 색상 상수가 같이 쓰여서 문자를 깜박이게 만든다.

참　　고　`textattr`,`textcolor`
정　　의　`conio.h`
예　　제　/* 문자의 색상을 노란색 깜박이는
　　문자로 설정한다.
　　*/
　　textcolor(YELLOW+BLINK);

쌇EOF
#상수 EOF

EOF는 보통 파일의 끝을 나타내는 문자로 사용된다. 아스키

문자로는 0x03 으로 정의된다. 이 상수는 주로 콘솔 함수의
에러를 나타내는 리턴 값으로 사용된다.

쌿COLORS
#상수 COLORS

COLORS 상수에는 CGA,EGA,VGA 그래픽 디스플레이 어댑터 에
서 사용하는 16색의 이름이 정의 되어 있다.

색상이름	상수
BLACK	0
BLUE	1
GREEN	2
CYAN	3
RED	4
MAGENTA	5
BROWN	6
LIGHTGRAY	7
DARKGRAY	8
LIGHTBLUE	9
LIGHTGREEN	10
LIGHTCYAN	11
LIGHTRED	12
LIGHTMAGENTA	13
YELLOW	14
WHITE	15

이상수는`conio.h`와`graphics.h`에 정의되어 있다.

쌿directvideo
#전역변수 DIRECTVIDEO

기　　능　비디오 출력을 제어하는 플래그 이다.
구　　문　extern int directvideo;
선언　파일　`conio.h`
기능　설명　프로그램의 콘솔출력 (예를들어`cputs`)이 비
디오 램으로 직접가는지 BIOS를 커처 가는지
를 결정한다.

디폴트 값은 1이 되어서 모든출력은 바이오스를 커치지 않고 화면에 출력된다. 값을 0으로 설정하면 모든 출력은 바이오스를 커치며 이렇게 되면 IBM BIOS 와 호환 시스템 에서만 사용할 수 있다.

쌇text_info
#파라미타 TEXT_INFO

텍스트 모드 화면에 관한 모든 정보를 포함하는 구조체 이다.`conio.h`에 정의되어 있다.`gettextinfo`함수도 참고 하기 바란다.

```
struct text_info {
      unsigned char winleft;        윈도 좌측끝
      unsigned char wintop;          윈도 상단끝
      unsigned char winright;       윈도 우측끝
      unsigned char winbottom;      윈도 하단끝
      unsigned char attribute;      문자 속성
      unsigned char normattr;       보통 속성
      unsigned char currmode;       현재 모드
      unsigned char screenheight; 화면 높이
      unsigned char screenwidth;   화면 넓이
      unsigned char curx;            커서좌표 x
      unsigned char cury;            커서좌표 y
};
```

쌇text_modes
#상수 TEXT_MODES

텍스트 모드를 나타내는 상수이다.`conio.h`에 정의 되어 있다.

기호상수명	수치	텍스트모드
LASTMODE	-1	이전의 텍스트 모드
BW40	0	흑백 40열
C40	1	칼라 40열
BW80	2	흑백 80열
C80	3	칼라 80열

MONO 7 단색(모노크롬) 80열

참 고 `textmode`

쌇ctype.h
#헤더파일 ctype.h

함수

`isascii` `isalunm` `isalpha`
`iscntrl` `isdigit` `isgraph`
`islower` `isprint` `ispunct`
`isspace` `isupper` `isxdigit`
`toascii` `_tolower` `tolower`
`_toupper` `toupper`

쌇isascii
#메크로함수 ISASCII

기 능 ASCII 문자 구분 매크로
구 문 #include <ctype.h>
int isascii(int c);
프로토타입 `ctype.h`
기능 설명 테이블에 의해서 ASCII코드로 된 정수값을 분류하는 메크로 이다. 이 함수는 문자 c가 기술 되는 항복을 검색한 후 참일 때는 0이외의 값을 리턴하고 거짓일 때는 0을 리턴한다.

isascii는 모든 정수값에 의해서 정의 된다.
리 턴 값 c의 하위 바이트가 0-127(0x00 - 0x7f) 사이에 있을때 0이외의 값이 리턴된다.
이 식 성 UNIX 시스템에서 사용할 수 있다.

쌇isalunm
#매크로함수 ISALUNM

기 능 문자 구분 매크로(영문 또는 숫자인지를 구분한다.)
구 문 #include <ctype.h>
inc isalnum(int c);
프로토타입 `ctype.h`

기능 설명 테이블에 의해서 ASCII코드로 된 정수값을 분류하는 메크로 이다. 이 함수는 문자 c가 기술 되는 항복을 검색한 후 참일 때는 0이외의 값을 리턴하고 거짓일 때는 0을 리턴한다.
리 턴 값 c가 문자(A-Z 혹은 a-z)이거나 숫자(0-9) 인 경우에 0이외의 값을 리턴한다.
이 식 성 UNIX 시스템에서 사용할 수 있다.

쌇isalpha
#매크로함수 ISALPHA

기 능 문자 구분 매크로
구 문 #include <ctype.h>
int isalpha(int c);
프로토타입 `ctype.h`
기능 설명 테이블에 의해서 ASCII코드로 된 정수값을 분류하는 메크로 이다. 이 함수는 문자 c가 기술 되는 항복을 검색한 후 참일 때는 0이외의 값을 리턴하고 거짓일 때는 0을 리턴한다.
리 턴 값 c가 문자(A-Z 혹은 a-z)인 경우에 0이외의 값을 리턴한다.
이 식 성 UNIX 시스템에서 사용할 수 있다.

쌇iscntrl
#매크로함수 ISCNTRL

기 능 콘트롤 문자 구분 매크로
구 문 #include <ctype.h>
int iscntrl(int c);
프로토타입 `ctype.h`
기능 설명 테이블에 의해서 ASCII코드로 된 정수값을 분류하는 메크로 이다. 이 함수는 문자 c가 기술 되는 항복을 검색한 후 참일 때는 0이외의 값을 리턴하고 거짓일 때는 0을 리턴한다.
리 턴 값 c가 삭제 문자이거나 일반적인 콘트롤 문자(0x7f 혹은 0x00-0x1f)인 경우에 0 이외의 값을 리턴한다.
이 식 성 UNIX 시스템에서 사용할 수 있다.

쌇isdigit
#매크로함수 ISDIGIT

기　　능　숫자 구분 매크로
구　　문　#include <ctype.h>
　　int isdigit(int c);
프로토타입　`ctype.h`
기능 설명　테이블에 의해서 ASCII코드로 된 정수값을 분
　　류하는 메크로 이다. 이 함수는 문자 c가 기
　　술 되는 항복을 검색한 후 참일 때는 0이외의
　　값을 리턴하고 거짓일 때는 0을 리턴한다.
리 턴 값　c가 숫자(0-9)일때만 0이외의 값을 리턴한다.
이 식 성　UNIX 시스템상에서 사용할 수 있으며 ANSI-C
　　와도 호환된다.

쌇isgraph
#매크로함수 ISGRAPH

기　　능　인쇄 가능한 문자 구분 매크로
구　　문　#include <ctype.h>
　　int isgraph(int c);
프로토타입　`ctype.h`
기능 설명　테이블에 의해서 ASCII코드로 된 정수값을 분
　　류하는 메크로 이다. 이 함수는 문자 c가 기
　　술 되는 항복을 검색한 후 참일 때는 0이외의
　　값을 리턴하고 거짓일 때는 0을 리턴한다.
리 턴 값　주어진 문자가 공백 문자를 제외한 인쇄 문자
　　일 경우에는 0 이외의 값을 리턴한다.
이 식 성　UNIX 시스템상에서 사용할 수 있으며 ANSI-C
　　와도 호환된다.

쌇islower
#매크로함수 ISLOWER

기　　능　소문자 구분 매크로
구　　문　#include <ctype.h>
　　int islower(int c);
프로토타입　`ctype.h`
기능 설명　테이블에 의해서 ASCII코드로 된 정수값을 분
　　류하는 메크로 이다. 이 함수는 문자 c가 기
　　술 되는 항복을 검색한 후 참일 때는 0이외의
　　값을 리턴하고 거짓일 때는 0을 리턴한다.
리 턴 값　c가 소문자(a-z)일 경우에 0이외의 값을 리턴

한다.
이 식 성 UNIX 시스템상에서 사용할 수 있으며 ANSI-C
와도 호환된다.

쌇isprint
#매크로함수 ISPRINT

기 능 인쇄 가능한 문자 구분 매크로
구 문 #include <ctype.h>
int isgraph(int c);
프로토타입 `ctype.h`
기능 설명 테이블에 의해서 ASCII코드로 된 정수값을 분
류하는 메크로 이다. 이 함수는 문자 c가 기
술 되는 항복을 검색한 후 참일 때는 0이외의
값을 리턴하고 거짓일 때는 0을 리턴한다.
리 턴 값 주어진 문자가 인쇄 문자(0x20-0x7f)일 경우
에는 0 이외의 값을 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있으며 ANSI-C
와도 호환된다.

쌇ispunct
#매크로함수 ISPUNCT

기 능 구두점 문자 구분 매크로
구 문 #include <ctype.h>
int ispunct(int c);
프로토타입 `ctype.h`
기능 설명 테이블에 의해서 ASCII코드로 된 정수값을 분
류하는 메크로 이다. 이 함수는 문자 c가 기
술 되는 항복을 검색한 후 참일 때는 0이외의
값을 리턴하고 거짓일 때는 0을 리턴한다.
리 턴 값 c가 구두점 문자 (iscntrl 혹은 isspace) 일
경우에 0이외의 값을 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있으며 ANSI-C
와도 호환된다.

쌇isspace
#매크로함수 ISSPACE

기 능 스페이스 공백 문자 구분 매크로
구 문 #include <ctype.h>

int isspace(int c);
프로토타입 `ctype.h`
기능 설명 테이블에 의해서 ASCII코드로 된 정수값을 분류하는 메크로 이다. 이 함수는 문자 c가 기술 되는 항복을 검색한 후 참일 때는 0이외의 값을 리턴하고 거짓일 때는 0을 리턴한다.
리 턴 값 c가 공백(스페이스), 탭, 복귀 문자, 개행 문자, 수직탭, 또는 Form Feed(0x09-0x0d,0x20)일 경우에는 0이외의 값을 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있으며 ANSI-C와도 호환된다.

쌇isupper
#매크로함수 ISUPPER

기 능 대문자 구분 매크로
구 문 #include <ctype.h>
int isupper(int c);
프로토타입 `ctype.h`
기능 설명 테이블에 의해서 ASCII코드로 된 정수값을 분류하는 메크로 이다. 이 함수는 문자 c가 기술 되는 항복을 검색한 후 참일 때는 0이외의 값을 리턴하고 거짓일 때는 0을 리턴한다.
리 턴 값 c가 대문자(A-Z)일 경우에 0이외의 값을 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있으며 ANSI-C와도 호환된다.

쌇isxdigit
#매크로함수 ISXDIGIT

기 능 16진법 문자 구분 매크로
구 문 #include <ctype.h>
int isxdigit(int c);
프로토타입 `ctype.h`
기능 설명 테이블에 의해서 ASCII코드로 된 정수값을 분류하는 메크로 이다. 이 함수는 문자 c가 기술 되는 항복을 검색한 후 참일 때는 0이외의 값을 리턴하고 거짓일 때는 0을 리턴한다.
리 턴 값 c가 16진수 문자인 경우에 0이외의 값을 리턴한다.

이 식 성 UNIX 시스템상에서 사용할 수 있으며 ANSI-C
와도 호환된다.

쌇toascii
#매크로함수 TOASCII

기 능 정수를 ASCII 문자로 변환한다.
구 문 int toascii(int c);
프로토타입 ctype.h
기능 설명 정수 c의 하위 7바이트를 제외하고 모두 소거
함으로써 값이 0에서 127사이에 있도록 한다.
이 함수는 정수 c를 ASCII로 변환하는 매크로
로 정의되어 있다.
리 턴 값 변환된 c의 값을 리턴한다.
이 식 성 UNIX 시스템 에서 사용할 수 있다.

쌇_tolower
#매크로함수 _TOLOWER

기 능 대문자를 소문자로 바꾼다.
구 문 #include <ctype.h>
int _tolower(int ch);
프로토타입 `ctype.h`
기능 설명 바꿀 문자 ch가 정확이 대문자인 경우에만 사
용할 수 있다는 점을 제외하면 tolower와 기
능이 같다.
_tolower를 사용하려면 반드시 ctype.h를 포
함 하여야 한다.
리 턴 값 ch가 대문자 이면 소문자로 바꾼다. 만약 대
문자가 아니라면 결과를 예상할 수 없다.
이 식 성 UNIX 시스템에서 사용할 수 있다.

쌇_toupper
#매크로함수 _TOUPPER

기 능 대문자를 소문자로 바꾼다.
구 문 #include <ctype.h>
int _toupper(int ch);
프로토타입 `ctype.h`
기능 설명 바꿀 문자 ch가 정확이 소문자인 경우에만 사
용할 수 있다는 점을 제외하면 toupper와 기

능이 같다.
_toupper를 사용하려면 반드시 ctype.h를 포함 하여야 한다.

리 턴 값 ch가 소문자 이면 대문자로 바꾼다. 만약 소문자가 아니라면 결과를 예상할 수 없다.

이 식 성 UNIX 시스템에서 사용할 수 있다.

쌇tolower

#함수 TOLOWER

기 능 대문자를 소문자로 바꾼다.

구 문 int tolower(int ch);

프로토타입 `ctype.h`

기능 설명 정수 ch를 소문자로 바꾼다 대문자가 아니면 변하지 않는다.

리 턴 값 ch가 대문자 이면 소문자를 리턴한다. 소문자이면 바뀌지 않는다.

이 식 성 UNIX 시스템에서 사용할 수 있다.

쌇toupper

#함수 TOUPPER

기 능 소문자를 대문자로 바꾼다.

구 문 int toupper(int ch);

프로토타입 `ctype.h`

기능 설명 정수 ch를 대문자로 바꾼다 소문자가 아니면 변하지 않는다.

리 턴 값 ch가 소문자 이면 대문자를 리턴한다. 대문자이면 바뀌지 않는다.

이 식 성 UNIX 시스템에서 사용할 수 있다.

쌇dir.h

#헤더파일 DIR.H

함수

`chdir` `findfirst` `findnext`
`fnmerge` `fnsplit` `getcurdir`
`getcwd` `getdisk` `mkdir`
`mktemp` `rmdir` `searchpath`
`setdisk`

상수, 데이타형, 파라미터

`ffblk`

쌇chdir
#함수 CHDIR

기　　능　현재의 디렉토리를 변경한다.
구　　문　int chdir(const char *path);
프로토타입　`dos.h`
기능 설명　path에 의해 지정된 디렉토리를 현재 작업중인 디렉토리로 변경할 수 있게 한다. path는 반드시 존재 하는 디렉로리 이어야 한다. 또 드라이브도 path 인수로서 지정될 수 있는데 ,chdir("a:\\turboc"); 은 지정된 현재 디렉토리 만을 변경하게 되며 작동중에 있는 드라이브는 변경하지 않는다.
리 턴 값　에러없이 수행되면 0을 리턴하고, 에러가 발생 되면 -1이 리턴된다. 이때`errno`는

ENOENT Path or file name not found

와 같이 설정된다.
이 식 성　UNIX 시스템 상에서 사용할 수 있다.
참　　조　`getcurdir`,`getcwd`,`mkdir`,`rmdir`, `system`

쌇findfirst
#함수 FINDFIRST

기　　능　디스크 디렉토리를 검색한다.
구　　문　#include <dir.h>
#include <dos.h>
int findfirst(cons char *pathname, struk ffblk *ffblk, int attrib);
프로토타입　`dir.h`
기능 설명　DOS 시스템 호출 0x4e를 이용해 디스크 디렉토리를 검색한다. pathname은 문자열로서 드라이브 지정, 경로지정, 찾고자 하는 파일명 등을 갖는다.

파일명 부분은 와일드 카드 (? 혹은 *)를 사용할 수 있다.
부합되는 파일이 발견되면 ffblk 구조체는 파일 디렉토리 정보로 채워진다.
구조체`ffblk`의 형식은 다음과 같다.

```
struct   ffblk   {
char      ff_reserved[21];/* 도스용 공간 */
char      ff_attrib;       /* 파일 속성 */
unsigned ff_ftime;         /* 파일 시간 */
unsigned ff_fdate;         /* 파일 날자 */
long      ff_fsize;        /* 파일 크기 */
char      ff_name[13];     /* 파일 이름 */
};
```

attrib 는 DOS파일 어트리뷰트 바이트로서 검색에 적합한 파일을 선택하는데 사용된다.
attrib 는`dos.h`에서 정의된 다음의 상수 중 하나 이다.

```
FA_RDONLY     읽기 전용 파일
FA_HIDDEN     숨겨진 파일
FA_SYSTEM     시스템 용 파일
FA_LABEL      볼륨 라벨
FA_DIREC      디렉토리
FA_ARCH       저장 속성 파일
```

findfirst는 DOS 디스크 전송 어드레스(DTA)를 ffblk의 어드레스로 설정한다. 이 DTA 값이 필요할 경우 findfirst를 호출한 뒤에 `getdta`나`setdta`를 사용해서 그것을 세이브하거나 복구 시켜야 한다.

리 턴 값 pathname의 검색과 일치하는 파일을 찾으면 0을 리턴하고 에러가 있으면 -1을 리턴한다.
이때 전역변수`errno`은 다음중 하나로 설정된다.

```
ENOENT    Path or file name not found
ENMFILE   No more file
```

이 식 성 DOS에 한한다.

참　　조 `findnext`

쌇findnext
#함수 FINDNEXT

기　　능 findfirst 검색을 계속한다.
구　　문 #include <dir.h>
　　int findnext(struct ffblk *ffblk);
프로토타입 `dir.h`
기능 설명 findfirst 에서 주어진 pathname 과 일치하는 다음 파일을 불러오는데 사용된다. ffblk 는 findfirst호출에 의해 같은 블록으로 채워진다. 이 블록은 검색을 계속하기 위해서 필요한 정보를 포함한다. findnext호출 때마다 한 개의 파일명이 리턴되며 pathname 과 일치하는 디렉토리나 파일명이 더 이상 발견되지 않을 때까지 사용할 수 있다.

findnext는 DOS 디스크 전송 어드레스(DTA)를 `ffblk`의 어드레스로 설정한다. 이 DTA 값이 필요할 경우`getdta`와`setdta`를 사용하여 그것을 저장하거나 복구하여야 한다.

리 턴 값 pathname과 일치하는 파일을 찾으면 0을 리턴한다. 더 이상의 파일명이 발견되지 않을때나 파일명에 에러가 있을경우 -1이 리턴된다. 이때 전역변수`errno`은 다음과 같이 설정된다.

ENOENT　Path or file name not found
ENMFILE　No more file

이 식 성 DOS에 한한다.
참　　조 `findfirst`

쌇fnmerge
#함수 FNMERGE

기　　능 구성 부분으로 패스를 구성한다.
구　　문 #include <dir.h>
　　void fnmerge(char *path,const char *drive,
　　const char *dir,
　　const char *name,
　　const char *ext);

프로토타입 `dir.h`
기능 설명 drive,dir,name,ext 로부터 path 파일명을 만
든다. 새로운 패스명을 C:\DIR\SUB\NAME.EXT
라고 한다. 이는 다음과 같이 구성된다.

drive = C
dir = \DIR\SUB
name = NAME
ext = EXT

fnmerge는 구성되는 패스명에 대해서 path안
에 충분한 공간이 있다는 것을 가정하고 있다
최대의 구성 길이는 dir.h에 정의된 MAXPATH
가 된다.

fnmerge와 fnsplit는 서로 반대의 성질을 가
지고 있다. fnsplit로 주어진 path를 분할할
경우, fnmerge로 부분 구성를 통합할 수 있고
path로 종료할 수 있게 된다.
리 턴 값 없음
이 식 성 DOS에 한한다.
참 조 `fnsplit`

쌇fnsplit
#함수 FNSPLIT

기 능 완전한 파일의 패스명을 각 구성 부분으로 분
할 한다.
구 문 #include <dir.h>
int fnsplit(const char *path,char *drive
char *dir,char *name,char *ext);
프로토타입 `dir.h`
기능 설명 파일의 완전한 패스명(path)를 X:\DIR\SUBDIR
\NAME.EXT 형태의 문자열로 취하면서 path를
4개의 구성 부분으로 분할한다. 이구성 부분
은 각각 drive,dir,name,ext 가 가르키는 문
자열에 저장한다.(5개의 구성 부분들이 모두
전달 되어야 하지만 이 중 어떤 것은 null이
될수도 있다. 이는 대응되는 구성 부분이 분
석은 되지만 저장되지는 않는다는 것을 의미
한다.

이들 문자열의 최대 크기는 dir.h에서 정의된 상수 MAXDIR,MAXPATH,MAXNAME,MAXMAEXT 에 의해서 주어진다. 그리고 각기 이들 크기는 null 종료를 위한 공간을 포함한다.

상수	(최대치)	문자열
MAXPATH	80	path
MAXDRIVE	3	drive;(콜론(:)을 포함
MAXDIR	66	dir;(시작과 끝에 역슬래쉬(\) 포함)
MAXNAME	9	name
MAXEXT	5	ext;(시작시 도트(.)를 포함)

fnsplit는 null로 되어 있지 않은 구성 부분을 저장할 수 있는 충분한 공간이 있다는 것을 전제하고 있다.

리 턴 값 path에서 완전한 패스명 구성 부분을 표시하는 정수를 리턴한다.(이것은 dir.h에서 정의된 5개의 플래크로 구성되어 있다) 이들 플래그와 구성 부분은 다음과 같이 나타난다.

EXTENSION An extension (ext가 존재하는 경우)
FILENAME A filename (name이 존재하는 경우)
DIRECTORY A directory (and possibly subdirectories) (dir이 존재하는 경우)
DRIVE A drive specification(see dir.h)(drive가 존재하는 경우
WILDCARDS Wildcards (* or ?) (와일드 카드가 존재하는 경우)

이 식 성 DOS 시스템 상에서 사용할 수 있다.
참 조 `fnmerge`

쌇getcurdir
#함수 GETCURDIR

기 능 지정된 드라이브를 현재 디렉토리로 불러 온다.
구 문 int getcurdir(int drive, char *directory);
프로토타입 `dir.h`
기능 설명 drive를 현재 작업중인 디렉토리로 불러온다. drive는 디렉토리 번호를 지정한다. (디폴트는 0 A 는 1..등). directory는 MAXDIR 길이의 메모리 영역을 가르킨다. 여기서는 null로 끌나는 디렉토리명이 위치하게 된다. 이때 드라이브 지정명을 포함하지도 않고 "\"로 시작하지도 않는다.
리 턴 값 에러가 없으면 0을 리턴하고 에러시에는 -1을 리턴한다.
이 식 성 DOS에 한한다.
참 조 `chdir`,`getcwd`,`getdisk`,`mkdir`,`rmdir`

쌿getcwd
#함수 GETCWD

기 능 작업중인 디렉토리명을 갖는다.
구 문 #include <dir.h>
char *getcwd(char *buf, int bufflen);
프로토타입 `dir.h`
기능 설명 작업중인 디렉토리의 완전한 경로명을 buffl-en 바이트의 buf에 저장한다. 전체 경로명에는 드라이브명도 포함된다.

전체 경로명이 bufflen보다 클경우(null포함)에는 에러가 발생하게 된다.
buf가 null이면`malloc`과 함께 bufflen의 길이 만큼 heap이 할당된다. 할당된 heap은 `free`를 통해 해제해 주어야 한다.
리 턴 값 getcwd는 다음을 리턴한다.

* buf가 입력상에서 null이 아닌경우
에러가 없으면 buf가 리턴되고, 에러시에는 null이 리턴된다.
* buf가 입력상에서 null인 경우
할당된 버퍼의 포인터를 리턴한다. 에러가 발생될때는 전역변수`errno`은 다음과 같이 설정된다.

ENODEV No such device 디바이스가 없다.
ENOMEM Not enough core 메모리 부족
ERANGE Result out of range 경로명이 bufflen의 길이
를 초과한다.

이 식 성 DOS 한한다.
참 조 `chdir`,`getcurdir`,`getdisk`,`mkdir`,
`rmdir`

쌇getdisk
#함수 GETDISK

기 능 현재의 드라이브 번호를 구한다.
구 문 int getdisk(void);
프로토타입 `dir.h`
기능 설명 현재의 드라이브를 정수값의 번호로 리턴한다
이는 도스호출 0x19와 같다. 리턴되는 정수값
과 실제 드라이브 와의 대응은 아래와 같다.

0 A 드라이브
1 B 드라이브
2 C 드라이브
3 D 드라이브

리 턴 값 현재 드라이브를 나타내는 번호를 리턴한다.
이 식 성 DOS에 한한다.
참 조 `getcurdir`,`getcwd`,`setdisk`

쌇mkdir
#함수 MKDIR

기 능 디렉토리를 만든다.
구 문 int mkdir(const char *path);
프로토타입 `dir.h`
기능 설명 주어진 경로명 path로부터 새로운 디렉토리를
만든다.
리 턴 값 새로운 디렉토리가 만들어져 있는 경우에 0을
리턴한다. 리턴값이 -1인 경우에는 에러를 나
타내고`errno`은 다음중 하나로 설정된다.

EACCES permission denied 엑세스 불가
ENOENT NO such file or directory 파일이나

디렉토리를 발견할 수 없다.
이 식 성 DOS에 한한다.
참 조 `chdir`,`getcurdir`,`getcwd`,`rmdir`

쌇mktemp
#함수 MKTEMP

기 능 단일 파일명을 만든다.
구 문 char *mktemp(char *template);
프로토타입 `dir.h`
기능 설명 template가 가르키는 문자열을 단일 파일명으로 대체하고 template를 리턴한다. template는 x가 6개 계속되는 null종료 분자열 이어야 한다.
6개의 x가 두개의 문자 그리고 피리어드와 세개의 문자로 변환되어 새로운 파일명이 된다. 파일명은 "AA.AAA" 로 부터 시작해서 새로운 파일명이 디스크에 있는지를 탐색해서 똑같은 파일명을 피하여 붙여진다.

리 턴 값 template가 옳바르게 만들어진 경우에 mktemp는 template문자열의 어드레스를 리턴하고 그렇지 않은경우에 null을 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있다.

쌇rmdir
#함수 RMDIR

기 능 DOS파일의 디렉토리를 삭제한다.
구 문 int rmdir(const char *path);
프로토타입 `dir.h`
기능 설명 path에서 주어진 디렉토리를 삭제한다.
이때 path로 지정된 디렉토리는

* 비어 있어야 하고
* 작업중인 디렉토리가 아니어야 하고
* 루트(root) 디렉토리가 아니어야 한다.

리 턴 값 디렉토리가 정상적으로 삭제되는 0을 리턴하고 에러가 발생되었을 때에는 -1을 리턴한다.
이때`errno`은 다음과 같이 설정된다.

EACESS 엑세스 불가
ENOENT 해당 파일혹은 디렉토리가 없다.
이 식 성 DOS에 한한다.
참 조 `chdir`,`getcurdir`,`getcwd`,`mkdir`

쌇searchpath
#함수 SEARCHPATH

기 능 파일에 대해 DOS path를 탐색한다.
구 문 char *searchpath(const char *file);
프로토타입 `dir.h`
기능 설명 문자열 PATH-...에 대한 DOS 경로를따라 file 을 찾는다. 완전한 경로명 문자열에 대한 포인터는 함수값으로 리턴된다.

searchpath는 현재 디렉토리에서 찾기 시작하여 환경변수 PATH에 있는 모든 디렉토리를 검색 한다. 만일 file 이 발견되면, 그 파일이 있는 완전한 경로명이 리턴되며, 그이름은 해당파일을 억세스 하는데 사용될 수 있다. (예를들면 fopen이나 exec...).

디렉토리명을 가진 문자열은 정적인 버퍼에 있으며 그 내용은 다음 searchpath 에 의해 변형된다.
리 턴 값 파일을 제대로 찾으면 그 파일이 있는 디렉토리의 이름에 대한 포인터를 리턴하지만, 찾지 못하면 null을 리턴한다.
이 식 성 DOS 에 한한다.
참 조 `exec`,`spawn`,`system`

쌇setdisk
#함수 SETDISK

기 능 현재 디스크 드라이브를 지정한다.
구 문 int setdisk(int drive);
프로토타입 `dir.h`
기능 설명 drive에 따라 현재 디스크 드라이브를 변경한다. drive 값은 A는0, B는1, C는2 와 같이 지정되어 있다. 이는 도스호출 0x0e와 동일하다
리 턴 값 사용 가능한 드라이브의 수를 리턴한다.

이 식 성 DOS에 한한다.
참 조 `getdisk`

쌿ffblk
#파라미타 FFBLK

ffblk 는 `findfirst`와`findnext`등의 함수에서 도스의 DTA를 대신하여 파일정보를 받는 장소로 사용된다.

```
struct ffblk {
  char          ff_reserved[21]; 도스용 공간
  char          ff_attrib;       파일 속성
  unsigned      ff_ftime;        파일 작성 시간
  unsigned      ff_fdate;        파일 작성 날자
  long          ff_fsize;        파일 크기
  char          ff_name[13];     파일 이름
};
```

쌿dos.h
#헤더파일 DOS.H

함수

`absread` `abswrite` `allocmem`
`bdos` `bdosptr` `country`
`ctrlbrk` `delay` `disable`
`dosexterr` `dostounix` `__emit__`
`enable` `FP_OFF` `FP_SEG`
`freemem` `geninterrupt` `getcwd`
`getdate` `getdfree` `getdta`
`getfat` `getfatd` `getftime`
`getpsp` `gettime` `getvect`
`getverify` `harderr` `hardresume`
`hardretn` `inp` `inport`
`inportb` `int86` `int86x`
`intdos` `intdosx` `intr`
`keep` `MK_FP` `nosound`
`outp` `outport` `outportb`
`parsfnm` `peek` `peekb`
`poke` `pokeb` `randbrd`
`randbwr` `segread` `setblock`

`setcbrk` `setdate` `setdta`
`settime` `setvect` `setverify`
`sleep` `sound` `unixtodos`
`unlink`

상수, 데이타형, 파라미터

`_8087` `_argc` `_argv`
`_heaplen` `_osmajor` `_osminor`
`_psp` `_doserrno` `environ`
`_stklen` `_version`

쌇absread
#함수 ABSREAD

기 능 특정 섹터를 읽는다.
구 문 int absread(int drive ,int nsects ,
int lsects ,void *buffer);
프로토타입 `dos.h`
기능 설명 어느 특정섹터를 읽는다. 디스크의 논리적 구조를 무시하고 파일이나 FAT 또는 디렉토리에도 유의하지 않는다. DOS 인터럽트 0x25를 통해 특정 디스크 섹터를 읽는다.

drive = 읽고자 하는 드라이브 번호
(0=A,1=B 등)
nsects = 읽고자 하는 섹터 번호
lsects = 논리적 섹터번호의 시작
buffer = 데이터가 읽혀지는 곳의 메모리 어드레스

읽고자 하는 섹터수는 buffer상의 세그먼트에 있는 메모리의 크기로 제한된다. 그러므로 64 KB는 absread를 사용하여 읽을수 있는 최대크기가 된다.

리 턴 값 에러가 없으면 0을 리턴한다. 에러가 발생하면 -1을 리턴하고`errno`에 시스템 호출에서 얻는 레지스터 AX의 값을 리턴한다.
이 식 성 DOS에 한한다.
참 조 `abswrite`,`biosdisk`

쌇abswrite
#함수 ABSWRITE

기　　능　절대 디스크 섹터를 쓴다.
구　　문　int abswrite(int drive ,int nsects,
　　int lsects ,void *buffer);
프로토타입　`dos.h`
기능 설명　어느 특정 디스크 섹터를 쓴다. 디스크의 논리적 구조를 무시하고 파일, FAT, 디렉토리에 유의 하지 않는다.
변칙적으로 사용될 경우 파일, 디렉토리, FAT에 겹쳐쓸 수도 있다.
DOS 인터럽트 0x26을 통해 특정 디스크 섹터를 쓴다.

drive = 쓰고자 하는 드라이브 번호(0=A,1=B 등)
nsects = 쓰고자 하는 섹터 번호
lsect = 논리적 섹터 번호의 시작
buffer = 데이타가 쓰여지는 곳의 메모리 어드레스

쓰고자 하는 섹터수는 buffer상의 세그먼트에 있는 메모리 크기로 제한된다. 그러므로 64KB는 abswrite를 한번에 호출하여 쓸수 있는 최대 메모리 크기이다.
리 턴 값　에러가 없으면 0을 리턴한다. 에러 발생시 -1을 리턴한다. 이때`errno`은 시스템호출 에서 리턴된 값을 리턴한다.
이 식 성　DOS에 한한다.
참　　조　`absread`,`biosdisk`

쌇allocmem
#함수 ALLOCMEM

기　　능　DOS 메모리 세그먼트를 할당한다.
구　　문　int allocmem(unsigned size,
　　unsigned *segp);
프로토타입　`dos.h`
기능 설명　사용할 수 있는 메모리 블록의 할당을 위해

DOS 시스템 호출 0x48을 사용하고 할당된 블록의 세그먼트 어드레스를 리턴한다.
size에는 확보된 메모리의 크기를 패러그래프 단위로 리턴한다.(하나의 패러 그래프는 16바이트 이다.)

segp는 새롭게 할당된 블록의 세그먼트 어드레스에 지정될 워드의 포인터이다. 사용 가능한 공간이 충분하지 않을경우 segp가 지정하는 워드에 대해 할당하지 않는다.
할당된 블록은 정렬된 패러그래프가 된다.

리 턴 값 에러가 없으면 -1을 리턴한다. 에러가 발생되면 사용가능한 메모리가 패러그래프 단위로 리턴 된다. allocmem 으로 부터의 에러리턴은 `_doserror`와`errno`를 다음과 같이 설정한다

ENOMEM Not enough memory 메모리 부족
이 식 성 DOS에 한한다.
참 조 `coreleft`,`freemem`,`malloc`,`setblock`

쌿bdos
#함수 BDOS

기 능 DOS 시스템을 호출한다.
구 문 int bdos(int dosfun, unsigned dosdx, unsigned dosal);
프로토타입 `dos.h`
기능 설명 bdos는 MS-DOS의 시스템 호출을 직접 엑세스하는 함수이다. 이와 같이 정수인수를 요하는 시스템 호출은 bdos를 사용한다. large 데이타 모델에서 호출 인수로서 포인터를 요하는 경우에 bdos 대신에 bdosptr을 사용하는 것이 중요하다.
dosfun은 시스템 호출의 번호이고 dosdx는 DX레지스터의 값이고 dosal은 AL레지스터의 값이다.
리 턴 값 bods의 리턴값은 시스템 호출에 의해서 설정된 AX의 값이다.
이 식 성 bdos는 도스에만 한정된다.
참 조 `bdosptr`,`geninterrupt`,`int86`,`int86x`,

`intdos`,`intdosx`

쌇bdosptr
#함수 BDOSPTR

기　　능　DOS 시스템을 호출한다.
구　　문　int bdosptr(int dosfun, void *argument,
　　unsigned dosal);
프로토타입　`dos.h`
기능 설명　DOS 시스템 호출을 직접 엑세스하는 기능을 제공한다. 포인터 인수를 요구하는 시스템 호출은 bdosptr을 사용한다.

large데이타 모델에 있어서 포인터 인수를 요하는 경우에는 bdosptr을 사용한다. small 데이타 모델에서는 bdosptr 에서는 argument 파라미터가 DX 레지스터를 지정하고, large 데이터 모델에서는 DS:DX로 주어진다.
dosal은 AL 레지스터의 값이다.
리 턴 값　bdosptr의 리턴값은 수행시에는 AX의 값이 되고 수행되지 않을 때는`errno`와`_doserror`가 설정된다.
이 식 성　DOS 에만 한정된다.
참　　조　`bdos`,`geninterrupt`,`int86`,`int86x`,
`intdos`,`intdosx`

쌇country
#함수 COUNTRY

기　　능　국가별 정보를 리턴한다.
구　　문　#include <dos.h>
struct country *country(int xcode,
struct country *cp);
프로토타입　`dos.h`
기능 설명　시간, 날자, 화폐 등과 같은 어떤 특정한 국가별 데이타를 어떻게 형식화 할 것인가를 결정한다. 이 함수에 의해서 설정되는 값은 DOS 버전에 따라 다르게 나타난다.

cp가 -1인 경우 현재국가의 xcode값이 0이 아닌 다른 값으로 설정되어 있어야 한다. cp 가

-1 이 아니면 cp 가 가르키는 구조체 country 가 현재국가 또는 xcode에 의해 주어진 국가 정보로 채워진다.

리 턴 값 에러없이 수행되면 country는 포인터 인수 cp 를 리턴하고 에러가 발생하면 null을 리턴한다.
이 식 성 DOS 버전 3.0 이상에서 사용 가능하다.

쌇ctrlbrk
#함수 CTRLBRK

기 능 <Ctrl+Break> 키를 눌렀을 때의 처리 루틴을 설정한다.
구 문 void ctrlbrk(int (*handler)(void));
프로토타입 `dos.h`
기능 설명 handler 에 의해 지정된 새로운 중단 처리 함수를 설정한다. 이때 지정된 함수는 직접적으로 실행되지는 않는다.

handler는 몇 번이고 시스템 호출과 작동을 수행할 수 있다. 처리루틴은 리턴값을 가질 필요가 없는데 이는 프로그램 상의 어느 위치이던 longjmp 를 사용해 리턴할 수 가 있기 때문이다. 처리함수가 0을 리턴하면, 프로그램은 이상 종료가 되며, 이외에는 프로그램이 재개된다.
리 턴 값 없음
이 식 성 DOS에 한한다.
참 조 `getcbrk`,`signal`

쌇delay
#함수 DELAY

기 능 일시적(약 1/1000초)으로 프로그램 실행을 중지 시킨다.
구 문 void delay(unsigned milliseconds);
프로토타입 `dos.h`
기능 설명 프로그램의 실행을 millisecons 가 지정한 만큼 중지 시킨다. 정확한 시간은 각기 오퍼레이팅 시스템에 따라서 달라질 수 있다.

리 턴 값 없음
이 식 성 IBM-PC와 호환 기종에서만 작동한다.
참 조 `nosound`,`sleep`,`sound`

쌿disable
#함수 DISABLE

기 능 인터럽트를 불가능 하게 한다.
구 문 #include <dos.h>
        void disable(void);
프로토타입 `dos.h`
기능 설명 프로그래머에게 융통성 있는 하드웨어 인터럽
        트 제어를 제공하도록 되어 있다. diable 매
        크로는 인터럽트를 불가능하게 하며 NMI 인터
        럽트 만이 외부 장치로부터 허용된다.
리 턴 값 없음
이 식 성 이 매크로는 8086 구조체에 한한다.
참 조 `enable`,`getvect`

쌿dosexterr
#함수 DOSEXTERR

기 능 확장된 DOS 에러의 정보를 설정한다.
구 문 #include <dos.h>
        int dosexterror(struct DOSERROR *eblkp);
프로토타입 `dos.h`
기능 설명 DOS 시스템 호출이 실패한 후 elblk가 가리키
        는 구조체 DOSERROR에 확장된 에러 정보를 리
        턴 한다. 이 구조체는 다음과 같이 정의된다.

        struct DOSERROR {
        int texterror; 화장된 에러
        char class; 에러 클래스
        char action; 동작
        char locus;
        };
리 턴 값 exterror 값을 리턴한다.
이 식 성 DOS 3.0 이하의 버전에서는 사용할 수 없다.

쌿dostounix
#함수 DOSTOUNIX

기 능 DOS 형식의 날짜와 시간을 UNIX 시간 형식으로 변환 시킨다.

구 문 #include <dos.h>
long dostounix(struct date *d,
struct time*t);

프로토타입 `dos.h`

기능 설명 `getdate`와`gettime`으로 받아들인 날짜와 시간을 UNIX 형식으로 변환한다. 여기서 d는 구조체 date를 가리키고 t는 구조체 time을 가리키는 포인터이다. 이들 구조체는 DOS 상의 날짜와 시간 정보를 가지고 있다.

리 턴 값 UNIX 형식의 현재의 날짜와 시간 파라미터 : 1970년 1월 1일 오정 0시(CMT) 이후부터 초까지 리턴한다.

이 식 성 DOS 에 한한다.

참 조 `unixtodos`

쌿__emit__

#함수 __emit__

기 능 인수의 값을 프로그램중에 직접 삽입한다.

구 문 void __emit__(argument,...);

프로토타입 `dos.h`

기능 설명 __emit__는 사용자로 하여금 목적코드가 컴파일 될때 직접 문자값을 삽입할 수 있도록 해주는 인라인 함수 이다.

따라서 이것은 인라인 어셈블리어나 어셈블러를 사용하지 않고 기계어 명령을 사용하는데 사용된다. 또한 이것은 인라인 어셈블리어를 사용할 수 없는 통합 개발 환경에서도 사용될 수 있다.

일반적으로 __emit__ 호출의 인수는 단일바이트 기계 명령어 이다. 그러나 이 함수의 기능으로 C 의 변수에 관해서 완벽하고 좀더 복잡한 명령을 수행할 수 있게 한다.

리 턴 값 없음

이 식 성 __emit__는 8086 계열에 한한다.

참 조

쌿enable
#함수 ENABLE

기　　능　하드웨어 인터럽트를 가능하게 한다.
구　　문　#unclude <dos.h>
　　void enable(void);
프로토타입　`dos.h`
기능 설명　하드웨어의 인터럽트 제어를 해제하고 인터럽
　　트가 가능하도록 해주는 매크로 이다.disable
　　에 의해 금지된 인터럽트를 해제하고 프로그
　　래머가 좀더 융통성 있게 하드웨어를 제어할
　　수 있도록 해준다.
리 턴 값　없음
이 식 성　8086 구조체에 한한다.
참　　조　`disable`,`getvect`

쌿FP_OFF
#함수 FP_OFF

기　　능　far 에드레스 오프셋을 구한다.
구　　문　#include<dos.h>
　　unsigned FP_OFF(farpointer);
프로토타입　`dos.h`
기능 설명　FP_OFF 매크로는 far 포인터인 farpointer 의
　　오프셋을 오프셋을 구하기 위해 사용될 수 있
　　다.
리 턴 값　FP_OFF는 오프셋값을 나타내는 unsigned 정수
　　값을 리턴한다.
참　　조　`FP_SEG`,`MK_FP`,`movedata`,`segread`

쌿FP_SEG
#함수 FP_SEG

기　　능　far 에드레스 세그먼트를 구한다.
구　　문　#include<dos.h>
　　unsigned FP_SEG(farpointer);
프로토타입　`dos.h`
기능 설명　FP_SEG 매크로는 far 포인터인 farpointer 의
　　세그먼트를 구하기 위해 사용될 수 있다.
리 턴 값　FP_SEG는 세그먼트를 나타내는 unsigned 정수

값을 리턴한다.
참 조 `FP_OFF`,`MK_FP`,`movedata`,`segread`

쌇freemem
#함수 FREEMEM

기 능 사전에 할당된 DOS 메모리 블록을 해제한다.
구 문 int freemem(unsigned segx);
프로토타입 `dos.h`
기능 설명 allocmem의 사전호출로 할당된 메모리 블록을 해제시킨다. segx는 블록의 세그먼트 어드레스 이다.
리 턴 값 에러없이 수행되면 0을 리턴한다. 에러 발생 시에는 -1을 리턴하며, 이때 errno는

ENOMEM Insufficient memory

와 같이 설정된다.
이 식 성 DOS 에 한한다.
참 조 `allocmem`,`free`

쌇geninterrupt
#함수 GENINTERRUPT

기 능 소프트웨어 인터럽트를 발생시킨다.
구 문 #include <dos.h>
void geninterrupt(int intr_num);
프로토타입 dos.h
기능 설명 geninterrupt 매크로는 intr_num에 의해 주어진 소프트웨어 인터럽트를 발생시킨다. 호출 후 모든 레지스터의 상탱는 호출된 인터럽트에 따라 다르다.
리 턴 값 없음
이 식 성 8086 구조체에 한한다.
참 조 `bdos`,`bdosptr`,`getvect`,`int86`, `int86x`,`intdos`,`intdosx`,`intr`

쌇getdate
#함수 GETDATE

기 능 시스템 일자를 지정한다.

구　　문　#include <dos.h>
　　void getdate(struct date *datep);
프로토타입　`dos.h`
기능 설명　datep가 가리키는 date구조체를 시스템의 현재 일자로 저장한다. 구조체 date는 다음과 같이 정의된다.

```
struct date {
int da_year;      현재 연
int da_day;       일자
int da_mon;       월
};
```

리 턴 값　없음
이 식 성　DOS 에 한한다.
참　　조　`ctime`,`gettime`,`setdate`,`settime`

쌿getdfree
#함수 GETDFREE

기　　능　디스크의 빈공간을 구한다.
구　　문　#include <dos.h>
　　void getdfree(unsigned char drive,
　　struct dfree *dtable);
프로토타입　`dos.h`
기능 설명　drive(디폴트 브라이브는 0 A=1...)에서 드라이브 지정을 받아서 dtable이 지정한 dfree 구조를 디스크의 특정 요소들로 저장한다.

dfree 구조는 다음과 같다.

```
struct dfree {
unsigned df_avail      사용가능 클러스터
unsigned df_total      총 클러스터
unsigned df_bsec       섹터당 바이트 수
unsigned df_sclus      클러스터당 섹터 수
};
```

리 턴 값　아무값도 리턴하지 않지만 에러발생 시 dfree 구조체의 df_sclus가 -1로 설정된다.
이 식 성　DOS 에 한한다.
참　　조　`getfat`,`getfatd`

쌿getdta

#함수 GETDTA

기　　능　디스크 전송 어드레스(DTA)를 구한다.
구　　문　char far *getdta(void);
프로토타입　`dos.h`
기능 설명　현재 설정된 DTA를 리턴한다. small과 medium
메모리 모델 에서는 세그먼트를 현재 데이터
세그먼트로 가정한다. C를 전용으로 사용하는
경우가 이런 경우이나 어셈블리 루틴은 DTA에
어떠한 하드웨어 어드레스도 설정할 수 있다.

compact, large, huge 메모리 모델 에서는
getdta에 의해 리턴되는 어드레스가 정확한
하드웨어 어드레스가 되며 프로그램 외부에
위치할 수 있다.
리 턴 값　현재 DTA를 가르키는 far 포인터를 리턴한다.
이 식 성　DOS에 한한다.
참　　조　`fcb`,`getdta`

쌇getfat
#함수 GETFAT

기　　능　지정된 드라이브의 파일 할당표(FAT)에 관한
정보를 구한다.
구　　문　#include <dos.h>
void getfat(unsigned char drive,
struct fatinfo *dtable);
프로토타입　`dos.h`
기능 설명　drive가 지정한 특정 드라이브의 파일 할당표
(FAT)에 관한 정보를 갖는다. 또한 dtable 은
fatinfo 구조체를 가르킨다. getfat 에 의해
채워지는 fatinfo 구조체는 다음과 같이 정의
된다.

struct fatinfo{
char fi_sclus;　　클러스터당 섹터수
char fi_fatid;　　FAT ID 바이트수
char fi_nclus;　　클러스터의 수
char fi_bytec;　　섹터당 바이트 수
};
리 턴 값　없음

이 식 성 DOS 에 한한다.
참 조 `getdfree`,`getfatd`

쌇getfatd
#함수 GETFATD

기 능 파일 할당표(FAT)에 관한 정보를 구한다.
구 문 #include <dos.h>
void getfatd(struct fatinfo *dtable);
프로토타입 `dos.h`
기능 설명 디폴트 드라이브의 파일 할당표(FAT)에 관한 정보를 갖는다. 또한 dtable 은 fatinfo 구조체를 가르킨다. getfat 에 의해 채워지는 fatinfo 구조체는 다음과 같이 정의 된다.

```
struct fatinfo{
char fi_sclus;        클러스터당 섹터수
char fi_fatid;        FAT ID 바이트수
char fi_nclus;        클러스터의 수
char fi_bytec;        섹터당 바이트 수
};
```

리 턴 값 없음
이 식 성 DOS 에 한한다.
참 조 `getdfree`,`getfat`

쌇getftime
#함수 GETFTIME

기 능 파일의 날짜와 시간을 갖는다.
구 문 #include <io.h>
int getftime(int handle,
struct ftime *ftimep);
프로토타입 `io.h`,`dos.h`
기능 설명 오픈되어 있는 handle과 연관된 디스크 파일에 대한 날짜와 시간을 갖는다. ftimep가 지정한 ftime 구조체는 시간과 일자로 채워지게 된다. ftime 구조는 다음과 같이 정의된다.

```
struct ftime {
 unsigned ft_tsec        초
 unsigned ft_min         분
 unsigned ft_hour        시
```

```
                unsigned ft_day             일
                unsigned ft_month           월
                unsigned ft_year            년 - 1980
               };
```
리 턴 값 에러가 없으면 0이 리턴되며, 에러가 발생하
면 -1이 리턴된다.
이 식 성 DOS에 한한다.
참 조 `open`,`setftime`

쌿getpsp
#함수 GETPSP

기 능 프로그램 세그먼트 prefix(PSP)를 구한다.
구 문 unsigned getpsp(void);
프로토타입 `dos.h`
기능 설명 도스 호출 0x62를 사용하여 PSP의 세그먼트
어드레스를 갖는다. 이 호출은 도스버전 3.X
이상 에서만 가능하다. 그 이하의 버전에서는
작동 코드에 의하여 설정되는 전역변수`_psp`
의 값이 대신 사용될 수 있다.
리 턴 값 PSP의 세그먼트 어드레스를 리턴한다.
이 식 성 DOS 3.X 에 한한다.
참 조 `getenv`,`_psp`

쌿gettime
#함수 GETTIME

기 능 시스템 시간을 구한다.
구 문 #include <dos.h>
void gettime(struct time *timep);
프로토타입 `dos.h`
기능 설명 timep가 가르키는 time 구조체에 시스템의 현
재 시간을 기입한다. time구조체는 다음과 같
이 정의된다.

```
struct time {
unsigned char ti_min;    분
unsigned char ti_hour;   시
unsigned char ti_hund;   100분의 1초
unsigned char ti_sec;    초
};
```

리 턴 값 없음
이 식 성 DOS에 한한다.
참 조 `getdate`,`setdate`,`settime`,`stime`,
`time`

쌇getvect
#함수 GETVECT

기 능 인터럽트 벡터를 구한다.
구 문 void interrupt(*getvect(int interruptno))();
프로토타입 `dos.h`
기능 설명 8086 계열의 모든 프로세서는 0부터 255까지의 번호가 매겨진 일련의 인터럽트 벡터를 가지고 있다. 각 벡터는 4바이트의 인터럽트 루틴의 어드레스를 가지고 있는 포인터 이다.

getvect는 interruptno에 의해 주어진 인터럽트 벡터값을 읽고 그것을 인터럽트 함수에 대한 far 포인터로서 리턴한다. interruptno 의 값은 0부터 255까지 될 수 있다.
리 턴 값 interruptno에 의해 지정된 인터럽트 벡터에 저장된 4바이트 값을 리턴한다.
이 식 성 DOS에 한한다.
참 조 `disable`,`enable`,`geninterrupt`,`setvect`

쌇getverify
#함수 GETVERIFY

기 능 DOS 의 verify flag 상태를 리턴한다.
구 문 int getverify(void);
프로토타입 `dos.h`
기능 설명 verify flag 의 현재 상태를 리턴하는데 verify flag는 디스크로의 출력을 제어한다. verify 가 off 이면 쓰기 작성은 검증되지 않는다 verify 가 on 일때 디스크의 쓰기작성이 검증된다.
리 턴 값 verify flag의 현재 상태(0이나 1)를 리턴한다.

리턴값 0 : verify flag off
리턴값 1 : verify flag on

이 식 성 DOS 에 한한다.
참 조 `setverify`

쌇harderr
#함수 HARDERR

기 능 하드웨어 에러 발생시 처리 함수를 정의한다.
구 문 void harderr(int (*handler)());
프로토타입 `dos.h`
기능 설명 프로그램에 대한 하드웨어 에러조정기(error handler)를 설정한다. 이 에러 조정기는 인터럽트 0x24가 발생될 때마다 호출된다. handler가 가르키는 이 함수는 다음과 같은 인수들이 전달된다.

handler(int errval, int ax,int bp,int si);
리 턴 값 없음
이 식 성 DOS에 한한다.
참 조 `hardresume`,`hardretn`,`peek`,`poke`

쌇hardresume
#함수 HARDRESUME

기 능 하드웨어 에러 조정기(error handler)
구 문 void hardresume(int axret);
프로토타입 `dos.h`
기능 설명 harderr에 의해 설정된 에러 조정기는 DOS 상태로 돌아가기 위해서 hardresume을 호출할 수 있다. 이 hardresume 의 결과 코드인 res code 의 리턴값은 중지(2), 재시도(1), 무시(0) 등의 지시자를 포함한다.

중지는 콘트롤 브레이크 인터럽트인 0x23 을 호출함으로써 수행된다. 이 처리기는 에러를 무시할 경우 0을 재시도할 경우에는 1을 그리고 중단하기 위해서는 2를 리톤해여 한다.
리 턴 값 없음
이 식 성 DOS에 한한다.
참 조 `harderr`,`hardretn`

쌇hardretn

#함수 HARDRETN

기　　능　하드웨어 에러 조정기(error handler)
구　　문　void hardretn(int retn);
프로토타입　`dos.h`
기능 설명　harderr에 의해 설정된 에러 조정기는 hardre
　　tn을 호출함으로써 응용 프로그램으로 직접
　　리턴할 수 있다. 이조정기는 무시할 경우(0)
　　재시도(1), 그리고 중단할 경우에는 (2)를
　　리턴해야 한다.
리 턴 값　없음
이 식 성　DOS에 한한다.
참　　조　`harderr`,`hardresume`

쌇inp
#함수 INP

기　　능　하드웨어 포트로부터 1개의 바이트를 읽어들
　　인다.
구　　문　include <dos.h>
　　int inp(int portid);
프로토타입　`dos.h`
기능 설명　portid가 지정한 포트로 부터 1 바이트를 읽
　　어 들인다.
리 턴 값　읽는값을 리턴한다.
이 식 성　8086 계열에 한한다.
참　　조　`inport`,`inportb`

쌇inport
#함수 INPORT

기　　능　하드웨어 포트로부터 1개의 워드를 읽어들인
　　다.
구　　문　include <dos.h>
　　int inport(int portid);
프로토타입　`dos.h`
기능 설명　portid가 지정한 포트로 부터 1 워드를 읽어
　　들인다.
리 턴 값　읽는값을 리턴한다.
이 식 성　8086 계열에 한한다.
참　　조　`inportb`,`outport`,`outportb`

쌇inportb
#함수 INPORTB

기　　능　하드웨어 포트로부터 1개의 바이트를 읽어들인다.
구　　문　include <dos.h>
　　int inportb(int portid);
프로토타입　`dos.h`
기능 설명　portid가 지정한 포트로 부터 1 바이트를 읽어 들인다.
리 턴 값　읽는값을 리턴한다.
이 식 성　8086 계열에 한한다.
참　　조　`inport`,`outport`,`outportb`

쌇int86
#함수 INT86

기　　능　일반 8086 소프트웨어 인터럽트
구　　문　#include <dos.h>
　　int int86(int intno, union REGS *inregs
　　union REGS *outregs);
프로토타입　`dos.h`
기능 설명　인수 intno에 의해 지정된 8086소프트웨어 인터럽트를 실행한다. 이것을 실행하기 전에 inregs로부터 설정한 레지스터값을 레지스터에 복사해 온다.
리 턴 값　int86은 인터럽트가 종료된 후에 AX 의 값을 리턴한다. 캐리 플래크가 0 이외의 값으로 살정되면 에러를 표시하고`_doserror`에 에러코드를 설정한다.
이 식 성　int86은 8086 계열에 한한다.
참　　조　`bdos`,`bdosptr`,`geninterrup`,`int86x`,
　　`intdo`,`intdosx`,`intr`

쌇int86x
#함수 INT86X

기　　능　일반 8086 소프트웨어 인터럽트 인터페이스
구　　문　#include <dos.h>
　　int int86x(int intno, union REGS *inregs,

union REGS *outregs,
struct SREGS *segregs);
프로토타입 `dos.h`
기능 설명 인수 intno 이 지정하는 소프트웨어 인터럽트를 호출한다.

리 턴 값 int86x는 인터럽트가 종료된 후에 AX 의 값을 리턴한다. 캐리 플래크가 0 이외의 값으로 살정되면 에러를 표시하고`_doserror`에 에러코드를 설정한다.
이 식 성 int86은 8086 계열에 한한다.
참 조 `bdos`,`bdosptr`,`geninterrupt`,`intdos`, `intdosx`,`int86`,`intr`,`sergread`

쌿intdos
#함수 INTDOS

기 능 일반 도스 인터럽트 인터페이스
구 문 #include <dos.h>
int intdos(union REGS *inregs,
union REGS *outregs);
프로토타입 `dos.h`
기능 설명 지정된 DOS 함수를 호출하도록 DOS 인터럽트 0x21을 호출한다.
리 턴 값 intdos는 인터럽트가 종료된 후에 AX 의 값을 리턴한다. 캐리 플래크가 0 이외의 값으로 살정되면 에러를 표시하고`_doserror`에 에러코드를 설정한다.
이 식 성 DOS 에 한한다.
참 조 `bdos`,`geninterrupt`,`int86`,`int86x`, `intdosx`,`intr`

쌿intdosx
#함수 INTDOSX

기 능 일반 도스 인터럽트 인터페이스
구 문 #include <dos.h>
int intdosx(union REGS *inregs,
union REGS *outregs,
struct SREGS *segregs);
프로토타입 `dos.h`

기능 설명 지정된 DOS 함수를 호출하도록 DOS 인터럽트 0x21을 호출한다.
리 턴 값 intdosx는 인터럽트가 종료된 후에 AX 의 값을 리턴한다. 캐리 플래크가 0 이외의 값으로 살정되면 에러를 표시하고`_doserror`에 에러코드를 설정한다.
이 식 성 DOS 에 한한다.
참 조 `bdos`,`geninterrupt`,`int86`,`int86x`, `intdos`,`intr`

쌇intr
#함수 INTR

기 능 8086 소프트웨어 인터럽트 인터페이스를 대체한다.
구 문 #include <dos.h>
void intr(int intno,struct REGPACK *preg);
프로토타입 `dos.h`
기능 설명 소프트웨어 인터럽트를 실행하기 위해 인터페이스를 대체시킨다. 이 함수는 인수 intno 가 지정하는 8086 소프트웨어 인터럽트를 발생시킨다. intr은 소프트웨어 인터럽트를 실행하기 전에 REGPACK 구조체 *preg 로부터의 레지스터값을 8086 레지스터로 복사한다.

소프트웨어 인터럽트가 완료된 후에는 intr은 플래그를 갖고 있으면 현재의 레지스터를 *preg로 복사한다.
리 턴 값 아무값도 리턴하지 않는다. REGPACK 구조체 *preg는 인터럽트 호출뒤의 레지스터 값을 갖는다.
이 식 성 8086 계열의 프로세서에 한한다.
참 조 `geninterrupt`,`int86`,`int86x`,`intdos`, `intdosx`

쌇keep
#함수 KEEP

기 능 프로그램의 상주 종료
구 문 void keep(unsigned char status, unsigned size);

프로토타입 `dos.h`
기능 설명 현재의 프로그램을 메모리에 상주시키고 종료 상태 status로 DOS에 제어를 리턴한다. 그러나 상주한 프로그램의 영역은 패러그래프 size 크기로 제한되고 그 나머지 영역은 해제 된다. keep은 TSR 프로그램을 만드는데 사용 될수 있다. keep은 도스함수 0x31을 사용한다
리 턴 값 없음
이 식 성 DOS에 한한다.
참 조 `abort`,`exit`

쌇MK_FP
#함수 MK_FP

기 능 far 포인터를 만든다.
구 문 #include <dos.h>
void far *MK_FP(unsigned seg
,unsigned ofs);
프로토타입 `dos.h`
기능 설명 세그먼트값 seg 와 오프셋값 ofs 로 부터 far 포인터를 만드는 매크로 이다.
리 턴 값 far 포인터를 리턴한다.
참 조 `FP_OFF`,`FP_SEG`,`movedata`,`segread`

쌇nosound
#함수 NOSOUND

기 능 PC 스피커의 기능을 중지 시킨다.
구 문 void nosound(void);
프로토타입 `dos.h`
기능 설명 sound 호출에 의해 스피커가 켜져있는 상태에서 스피커의 기능을 중지 시킨다.
리 턴 값 없음
참 조 `delay`,`nosound`

쌇outp
#함수 OUTP

기 능 하드웨어 포트로 바이트를 출력한다.
구 문 void outb(int portid, int value);
프로토타입 `dos.h`

기능 설명 portid 가 지정한 포트로 value의 값을 출력
한다.
리 턴 값 없음
이 식 성 8086 계열에 한한다.
참 조 `inport`,`inportb`,`outport`

쌿outport
#함수 OUTPORT

기 능 하드웨어 포트로 워드를 출력한다.
구 문 void outport(int portid, int value);
프로토타입 `dos.h`
기능 설명 portid 가 지정한 포트로 value의 값을 출력
한다.
리 턴 값 없음
이 식 성 8086 계열에 한한다.
참 조 `inport`,`inportb`,`outportb`

쌿outportb
#함수 OUTPORTB

기 능 하드웨어 포트로 바이트를 출력한다.
구 문 void outportb(int portid, int value);
프로토타입 `dos.h`
기능 설명 portid 가 지정한 포트로 value의 값을 출력
한다.
리 턴 값 없음
이 식 성 8086 계열에 한한다.
참 조 `inport`,`inportb`,`outport`

쌿parsfnm
#함수 PARSFNM

기 능 파일명을 분석한다.
구 문 #include <dos.h>
char *parsfnm(const char *smdline,
struct fcb *fcb, int opt);
프로토타입 `dos.h`
기능 설명 파일 이름용 cmdline이 가리키는 문자열을 분
석한다. 이 문자열은 정상적인 명령어행이다.
파일명은 드라이브, 파일명, 그리고 확장자로

FCB에 놓여진다. 이 FCB는 fcb에 의해 설정된다.

opt 는 DOS 분석 시스템 호출에서 AL에 대한 값이다.

리 턴 값 에러가 없으면 파일명의 마지막 문자 다음에 오는 바이트의 포인터를 리턴한다. 에러 발생시 null을 리턴한다.

이 식 성 DOS 에 한한다.

쌇peek

#함수 PEEK

기 능 segment : offset 에 의하여 지정된 메모리의 위치의 1워드를 리턴한다.

구 문 int peek(unsigned segment,
unsigned offset);

프로토타입 `dos.h`

기능 설명 segment:offset 으로 지정되는 메모리 위치의 1 워드를 리턴한다. dos.h가 포함될 때 peek 가 호출되면 인라인 코드로 확장되는 매크로로 처리 된다. 반대로 dos.h와 peek를 #undef 로 했을경우 함수로 취급된다.

리 턴 값 segment : offset 의 메모리 위치에 저장된 1 워드를 리턴한다.

이 식 성 8086 계열에 한한다.

참 조 `harderr`,`peekb`,`poke`

쌇peekb

#함수 PEEKB

기 능 segment : offset 에 의하여 지정된 메모리의 위치의 1바이트를 리턴한다.

구 문 int peekb(unsigned segment,
unsigned offset);

프로토타입 `dos.h`

기능 설명 segment:offset 으로 지정되는 메모리 위치의 1 바이트를 리턴한다. dos.h가 포함될때 peek 가 호출되면 인라인 코드로 확장되는 매크로로 처리 된다. 반대로 dos.h와 peek를 #undef 로 했을경우 함수로 취급된다.

리 턴 값 segment : offset 의 메모리 위치에 저장된 1 바이트를 리턴한다.
이 식 성 8086 계열에 한한다.
참 조 `harderr`,`peek`,`pokeb`

쌇poke
#함수 POKE

기 능 segment:offset 으로 지정된 위치에 1 워드를 저장한다.
구 문 void port(unsigned segment,
unsigned offset, int value);
프로토타입 `dos.h`
기능 설명 segment:offset으로 표시된 메모리 위치에 1 워드를 저장한다. dos.h 가 포함될때 poke 루틴은 매크로로 처리된다. 반면 dos.h 를 포함 시키지 않거나 #undef poke를 포함시키면 함수로 처리된다.
리 턴 값 없음
이 식 성 8086 계열에 한한다.
참 조 `harderr`,`peek`,`pokeb`

쌇pokeb
#함수 POKEB

기 능 segment:offset으로 표시된 메모리 위치에 1 바이트를 저장한다.
구 문 void portb(unsigned segment,
unsigned offset,byte value);
프로토타입 `dos.h`
기능 설명 segment:offset으로 표시된 메모리 위치에 1 바이트를 저장한다. dos.h 가 포함될때 poke 루틴은 매크로로 처리된다. 반면 dos.h 를 포함 시키지 않거나 #undef pokeb를 포함시키면 함수로 처리된다.
리 턴 값 없음
이 식 성 8086 계열에 한한다.
참 조 `harderr`,`peek`,`poke`

쌇randbrd
#함수 RANDBRD

기　　능　랜덤 블록을 읽는다.
구　　문　#include <dos.h>
　　int randbrd(struct fcb *fcb,int rcnt);
프로토타입　`dos.h`
기능 설명　fcb가 지정한 파일 콘트롤 블록(FCB)을 사용
　　해서 rcnt 개의 레코드를 읽는다. 레코드는
　　현재의 디스크 전송 어드레스(DTA) 에 있는
　　메모리를 사용해서 읽는다.

　　이들은 FCB의 random record field에서 지정
　　한 디스크 레코드로부터 읽혀진다. 이것은 도
　　스 시스템 호출 0x27을 호출하여 수행된다.

　　읽어 들인 실제 레코드수는 FCB의 랜덤 레코
　　드 필드를 조사하여 결정된다.

리 턴 값　연산의 결과의 따라서 다음 값이 리턴된다.
　　0 모든 레코드가 읽혀졌다.(정상종료)
　　1 파일 종료(EOF)에 이르러서 마지막 레코드
　　를 완전하게 읽어들였다.
　　2 레코드의 읽기가 0xffff번지를 넘었다.(많
　　은 레코드가 읽혀지고 있다.)
　　3 마지막 레코드가 종료되지 않은 상태에서
　　파일 종료에 이르렀다.

이 식 성　DOS 에 한한다.
참　　조　`getdta`,`randbwr`,`setdta`

쌇randbwr
#함수 RANDBWR

기　　능　랜덤 블록을 쓴다.
구　　문　#include <dos.h>
　　int randbwr(struct fcb *fcb,int rcnt);
프로토타입　`dos.h`
기능 설명　fcb가 가리키는 FCB를 사용해서 디스크에 rcn
　　t 개의 레코드를 써넣는다.
　　이것은 도스 시스템 호출 0x28을 호출하여 수
　　행된다. rcnt 가 0이면 그 파일은 랜덤 레코
　　드 필드가 지정한 길이에 따라 잘려지게 된다

쓰여진 실제 레코드수는 FCB의 random record field 를 조사해서 결정될수 있다.
랜덤 레코드 필드는 쓰여진 레코드수에 선행된다.

리 턴 값 연산의 결과에 따라 다음 값들이 리턴된다.
0 모든 레코드들이 쓰여졌다. (정상종료)
1 레코드를 써넣을 디스크 공간이 부족하다.
2 레코드를 써넣은 것이 0xffff 번지를 넣었다. (많은 레코드들이 사용되고 있다.)
이 식 성 randbwr은 DOS에 한한다.
참 조 `randbrd`

쌇segread
#함수 SEGREAD

기 능 세그먼트 레지스터를 읽는다.
구 문 #include <dos.h>
void segread(struct SREGS *segp);
프로토타입 `dos.h`
기능 설명 segp가 지정한 위치의 구조체에 현재의 세그먼트 레지스터 내용을 저장한다. 이 기능은 intdosx,int86x와 함께 사용하기 위한것이다.
리 턴 값 없음
이 식 성 8086 구조체에 한한다.
참 조 `FP_OFF`,`int86`,`intdos`,`MK_FP`, `movedata`

쌇setblock
#함수 SETBLOCK

기 능 사전 할당된 블록의 크기를 변경한다.
구 문 int setblock(unsigned segx,
unsigned newsize);
프로토타입 `dos.h`
기능 설명 메모리 세그먼트 크기를 변경한다. segx는 이전의 allocmem을 호출하여 얻은 세그먼트 어드레스 이다. newsize는 패러그래프 단위의 새로운 크기이다.
리 턴 값 에러가 없으면 -1 을 리턴한다. 에러 발생시 가능한 최대 블록의 크기를 패러그래프 단위

로 리턴하고`_doserrno`를 설정한다.
이 식 성 DOS에 한한다.
참 조 `allocmem`

쌇setcbrk
#함수 SETCBRK

기 능 control-break 검사를 설정한다.
구 문 int setblock(int cbrkvalue);
프로토타입 `dos.h`
기능 설명 control-break 검사를 on이나 off하기 위하여
DOS 시스템 호출 0x33을 이용한다.

value = 0 검사를 off 한다. (콘솔, 프린터,
통신 장치의 IO 때만 검사한다.)
value = 1 검사를 한다. (모든 시스템 호출시
검사한다.)
리 턴 값 전달된 값 cbrkvalue를 리턴한다.
이 식 성 DOS 에 한한다.
참 조 `getcbrk`

쌇setdate
#함수 SETDATE

기 능 DOS에 날짜를 설정한다.
구 문 #include <dos.h>
void setdate(struct date *datep);
프로토타입 `dos.h`
기능 설명 datep가 지정한 위치의 date 구조체에 있는대
로 시스템 날짜(월,일,년)을 지정한다.
date 구조체는 아래와 같다.

struct date {
int da_year; 년
int da_day; 월
int da_mon; 일
};
리 턴 값 없음
이 식 성 DOS에 한한다.
참 조 `getdate`,`gettime`,`settime`

쌿setdta
#함수 SETDTA

기　　능　디스크 전송 어드레스를 설정한다.
구　　문　void setdta(char far *dta);
프로토타입　`dos.h`
기능 설명　DOS 의 DTA(disk transfer address)를 dta 로
　　바꾼다.
리 턴 값　없음
이 식 성　DOS 에 한한다.
참　　조　`getdta`

쌿settime
#함수 SETTIME

기　　능　시스템 시간을 설정한다.
구　　문　#include <dos.h>
　　void settime(struct time *timpe);
프로토타입　`dos.h`
기능 설명　timep가 설정한 위치의 time 구조체의 내용대
　　로 시스템 시간을 설정한다.
　　time 구조체의 내용은 다은과 같다.

　　struct time {
　　unsigned char ti_min;　　분
　　unsigned char ti_hour;　　시
　　unsigned char ti_hund;　　100분의 1초
　　unsigned char ti_sec;　　초
　　};
리 턴 값　없음
이 식 성　DOS에 한한다.
참　　조　`ctime`,`getdate`,`gettime`,`setdate`,
　　`time`

쌿setvect
#함수 SETVECT

기　　능　인터럽트 벡터를 설정한다.
구　　문　void setvect(int interruptno,
　　void interrup(*isr)());
프로토타입　`dos.h`

기능 설명 8086 계열의 모든 프로세서는 0부터 255까지의 번호가 매겨진 일련의 인터럽트 벡터를 가지고 있다. 각 벡터는 4바이트의 인터럽트 루틴의 어드레스를 가지고 있는 포인터 이다.

setvect는 interruptno 에 의해 지명된 인터럽트 벡터의 값을 far 포인터 isr로 바꾼다. C루틴의 어드레스는 그 루틴이 interrupt routine 으로 선언되었을 때만 isr로 전달될 수 있다.

만일 사용자가 dos.h에서 선언된 함수 원형을 사용 한다면 어떠한 메모리 모델에서도 함수의 어드레스를 간단하게 전달할 수 있다.

리 턴 값 없음
이 식 성 8086 계열의 프로세서에 한한다.
참 조 `getvect`

쌿setverify
#함수 SETVERIFY

기 능 DOS의 확인 플래그 상태를 설정한다.
구 문 void setverify(int value);
프로토타입 `dos.h`
기능 설명 현재의 verify 프래그를 value 값으로 설정한다.
value 0 = verify off
value 1 = verify on

verify플래그는 디스크로의 출력을 검증한다. 값이 0일 때에는 검증을 하지 않으며 값이 1일 때에는 출력을 행할때마다 검증을 하게된다.

리 턴 값 없음
이 식 성 DOS에 한한다.
참 조 `getverify`

쌿sleep
#함수 SLEEP

기 능 일정 시간(초단위) 동안 실행을 중지한다.

구　　문　void sleep(unsigned seconds);
프로토타입　`dos.h`
기능 설명　sleep를 호출하면 프로그램은 seconds초 만큼 정지한다. 시간의 정확도는 100분의 1초 혹은 정확도가 약간 떨어지는 DOS clock 의 정확도와 같다.
리 턴 값　없음
이 식 성　UNIX 시스템 상에서 사용할 수 있다.
참　　조　`delay`

쌇sound
#함수 SOUND

기　　능　지정된 주파수로 PC 스피커가 소리를 내게한다.
구　　문　void sound(unsigned frequency);
프로토타입　`dos.h`
기능 설명　sound 는 frequency 의 주파수로 PC 스피커가 소리를 내도록 한다. frequency는 Hertz(초당 주파수)로 표현된다. 스피커의 소리를 끝 내려면 nosound 를 호출하여야 한다.
이 식 성　IBM-PC와 그 호환기종 에서만 작동된다. 동일한 기능이 터보 파스칼에도 있다.
리 턴 값　`delay`,`nosound`

쌇unixtodos
#함수 UNIXTODOS

기　　능　날짜와 시간을 DOS 포맷으로 변환한다.
구　　문　#include <dos.h>
void unixtodos(long time,struct date *d,
struct time *t);
프로토타입　`dos.h`
기능 설명　time에 주어진 UNIX 포맷의 날짜, 시간정보를 도스포맷 으로 바꾸어서 d와 t가 가리키는 date 와 time 구조체에 저장한다.
리 턴 값　없음
이 식 성　DOS에 한한다.
참　　조　`dostounix`

쌇unlink

#함수 UNLINK

기　　능　파일을 삭제한다.
구　　문　int unlink(const char *filename);
프로토타입　`dos.h`,`io.h`,`stdio.h`
기능 설명　filename 으로 표시된 파일을 삭제한다. 어떠한 DOS 장치, 패스, 파일 명도 filename 으로 쓸 수 있다. 와일드 카드는 쓸 수 없다.
읽기 전용파일은 함부로 삭제할 수 없다. 읽기 전용파일을 삭제하기 위해서는 먼저`chmod`나`_chmod`를 호출하여 읽기 전용의 속성을 바꾸어야 한다.

리 턴 값　에러가 없으면 0을 리턴한다. 에러를 발생하면 -1을 리턴하며`errno`는 다음중 하나로 설정 된다.

ENOENT　경로나 파일을 찾을 수 없음
EACCES　억세스 불가

이 식 성　UNIX 시스템 상에서 사용될 수 있다.
참　　조　`chmod`,`remove`

쌇_8087
#전역변수 _8087

기　　능　Coprocessor 칩 플래그이다.
구　　문　extern int _8087;
선언 파일　`dos.h`
기능 설명　기동코드 자동탐지논리가 부동소숫점 coprocessor 를 탐지하는 경우에는 _8087 변수가 0이 아닌 값(1,2,3)으로 설정되고 그렇지 않은 경우에는 0으로 설정된다.

쌇_argc
#전역변수 _ARGC

기　　능　명령어행 인수의 수를 갖는다.
구　　문　extern int _argc;
선언 파일　`dos.h`
기능 설명　프로그램이 실행될 때 메인 함수에 argc 값을 제공한다.

쌇_argv
#전역변수 _ARGV

기　　능　명령어행 인수에 대한 포인터를 배열한다.
구　　문　extern char *_argv[];
선언 파일 `dos.h`
기능 설명　프로그램이 시작될 때 메인 함수에 제공되는
　　초기 명령어 행의 인수 (argv[]의 원소) 를
　　포함하는 배열을 가리킨다.

쌇_heaplen
#전역변수 _HEAPLEN

기　　능　near heap의 길이를 갖는다.
구　　문　extern unsigned _heaplen;
선언 파일 `dos.h`
기능 설명　small 데이타 모델에서 near 의 크기를 지정
　　한다. _heaplen은 large 데이타 모델(compac-
　　t, large, huge) 모델에서는 존재 하지 않는
　　데 이유는 이들이 near heap 을 갖지 않기 때
　　문 이다.

쌇_osmajor
#전역변수 _OSMAJOR,_OSMINOR

기　　능　major 와 minor의 DOS 버전 번호를 포함한다
구　　문　extern unsigned char _osmajor;
　　extern unsigned char _osminor;
선언 파일 `dos.h`
기능 설명　major와 minor 버전 번호는 개별적으로 _osma
　　jor와 _osminor를 통해서 사용될 수 있다.

쌇_osminor
#전역변수 _OSMAJOR,_OSMINOR

기　　능　major 와 minor의 DOS 버전 번호를 포함한다
구　　문　extern unsigned char _osmajor;
　　extern unsigned char _osminor;
선언 파일 `dos.h`
기능 설명　major와 minor 버전 번호는 개별적으로 _osma

jor와 _osminor를 통해서 사용될 수 있다.

쌇_psp
#전역변수 _PSP

기　　능　프로그램 세그먼트 프리픽스를 포함한다.
구　　문　extern unsigned int _psp;
선언 파일 `dos.h`
기능 설명　현재 프로그램을 위해서 세그먼트 프리픽스 (PSP)의 세그먼트 어드레스를 포함한다. PSP는 DOS가 필요한 프로그램의 정보를 가지고 있다.

쌇_doserrno
#전역변수 _DOSERRNO

기　　능　도스 호출 에러를 갖는다.
구　　문　extern int _doserrno
선언 파일 `dos.h`,`errno`
기능 설명　DOS 시스템 호출이 에러가 날때 _doserrno은 실제 DOS 에러코드로 설정된다. errno은 UNIX 시스템 에서 비롯된 같은 종류의 에러 변수이다.

기 호	DOS 에러 코드
EINVAL	Bad function 잘못된 변수
E2BIG	Bad environ 잘못된 환경
EACCES	Access denied 잘못된 억세스
EBADF	Bad handle handle 값이 잘못됨
EFAULT	Reserved 예약어
EINVAL	Bad data 잘못된 데이터
EMFILE	Too many open 너무 많이 열려 있음.
ENOENT	No such file or directory 지정한 파일이나 디렉토리가 없음
ENOEXEC	Bad format 잘못된 포맷
ENOMEM	Bad block Mcb 파괴됨
ENOMEM	Out of memory 메모리 부족
ENOMEM	Bad block 블록이 틀림
EXDEV	Bad drive 드라이브가 틀림

EXDEV Not same device 같은 디바이스가 아님

쌇environ
#전역변수 ENVIRON

기 능 DOS 환경 변수를 억세스한다.
구 문 extern char *environ[];
선언 파일 `dos.h`
기능 설명 문자열을 가리키는 포인터 배열로서 DOS 환경 변수의 변경과 억세스에 사용된다. 각 문자열의 형식은 envar=varvalue 이다.

여기에서 envar는 환경변수의 이름(PATH와 같이)이고 varvalue는 envar에 설정되어 있는 문자열의 값(C:\BIN\;C:\DOS 와 같이)이다.

쌇_stklen
#전역변수 _STKLEN

기 능 스택의 크기를 갖는다.
구 문 extern unsigned _stklen;
선언 파일 `dos.h`
기능 설명 6개 메모리 모델에 대한 스택의 크기를 지정한다. 최소 스택 크기는 128워드만 허용되지만 사용자가 더 작은 값을 설정하면 _stklen은 자동적으로 최소로 조정된다. 디폴트 스택의 크기는 4K 이다.

쌇_version
#전역변수 _VERSION

기 능 DOS 버전 번호를 갖는다.
구 문 extern unsigned int _version;
선언 파일 `dos.h`
기능 설명 _version은 DOS 버전 번호를 갖는데 major 버번 번호는 low 바이트로 minor 버전 번호는 high 바이트로 된다. 예를 들어 DOS 버전 x,y 중 x는 major 버전 번호이고 y는 minor 이다.

쌇errno.h

#헤더파일 ERRNO.H

상수, 데이타, 파라미터

`_doserrno`
`errno`

쌇errno
#헤더파일 ERRNO

기　　능　시스템 에러를 지정한다.
구　　문　extern int errno;
선언　파일　`errno.h`
기능　설명　시스템 호출에서 에러 발생시 errno는 에러의 형태를 지정하도록 설정된다. 때때로 errno와 _doserrno는 같은 기능을 하지만 errno는 dos errno에 포함된 실제 DOS 에러를 포함하지 않는 경우도 있다. 이 경우 _doserrno가 아닌 errno만을 설정한 에러가 발생하게 된다.

기 호	에러 코드
E2BIG	Arg list too long 인수 리스트가 너무 많음
EACCES	permission denied 억세스가 허용되지 않음
EBADF	Bad file number 파일 번호가 틀림
ECONTR	Memory Block destroied 메모리 블록이 파괴되었음
ECURDIR	Attemp to remove CurDir 현재의 디렉토리를 삭제하려고 함
EDOM	Domain error 인수가 범위를 초과함
EEXIST	file already exist 파일이 이미 존재함
EINVACC	Invalid access code 억세스 코드가 잘못됨
EINVAL	Invalid argument 인수가 잘못됨
EINDAT	Invalid data 데이터가 잘못됨
EINVDRV	Invalid drive specified 드라이브를 잘못 지정
EINVENV	Invalid evironment 잘못된 환경
EINVFMT	Invalid format 잘못된 포맷
EINVFNC	Invalid function number 잘못된 함수 번호
EINVMEM	Invalid memory block address 잘못된 메모

리 블록 어드레스
EMFILE Too many open file 너무 많은 파일이 열려 있음
ENODEV No such device 지정한 디바이스가 없음
ENOENT No such file or directory 지정한 파일이나 디렉토리가 없음
ENOEXEC Exec format error EXEC 파일의 형식이 틀림
ENOFILE No such file or directory 지정한 파일이나 디렉토리가 없음
ENOMEM Not enough memory 메모리 부족
ENOPATH Path not found 패스명 지정이 잘못 되었음
ENOTSAM Not same device 같은 디바이스가 아님
ERANGE Result out of range 결과가 범위를 초과
EXDEV Cross device link 크로스 디바이스 링크
EZERO 0 에러없음

쌿fcntl.h
#헤더파일 fcntl.h

헤더파일 fcntl.h 에서는 파일을 열기위한 모드 상수가 제공 된다.

`O_APPEND` `O_RDWR`
`O_BINARY` `O_TEXT`
`O_CREAT` `O_TRUNC`
`O_EXCL` `O_WRONLY`
`O_RDONLY`

쌿O_APPEND
#엑세스 플래그

O_NDELAY UNIX 호환용, 여기서는 사용되지 않는다.
O_RDONLY 읽기 전용으로 개방
O_WRONLY 쓰기 전용으로 개방
O_RDWR 읽기와 쓰기용으로 개방
O_APPEND 이 플래그가 설정된 경우 파일 포인터가 파일끝으로 설정된다.
O_CREAT 이 플래그는 파일이 존재할 경우 영향이 없다.
파일이 존재 하지 않을 경우 파일이 만들어 지고
mode 비트가 파일의 속성을 결정하는데 사용된다

O_TRUNC 파일이 존재하는 경우 파일의 크기는 0 으로 되지만, 파일의 속성은 변경되지 않는다.
O_EXCL O_CREAT 와 같이 사용된다. 파일이 존재하는 경우 에러가 리턴된다.
O_BINARY 파일을 이진 모드로 개방한다.
O_TEXT 파일을 텍스트 모드로 개장한다.

쌇O_BINARY
#엑세스 플래그

O_NDELAY UNIX 호환용, 여기서는 사용되지 않는다.
O_RDONLY 읽기 전용으로 개방
O_WRONLY 쓰기 전용으로 개방
O_RDWR 읽기와 쓰기용으로 개방
O_APPEND 이 플래그가 설정된 경우 파일 포인터가 파일끝으로 설정된다.
O_CREAT 이 플래그는 파일이 존재할 경우 영향이 없다. 파일이 존재 하지 않을 경우 파일이 만들어 지고 mode 비트가 파일의 속성을 결정하는데 사용된다
O_TRUNC 파일이 존재하는 경우 파일의 크기는 0 으로 되지만, 파일의 속성은 변경되지 않는다.
O_EXCL O_CREAT 와 같이 사용된다. 파일이 존재하는 경우 에러가 리턴된다.
O_BINARY 파일을 이진 모드로 개방한다.
O_TEXT 파일을 텍스트 모드로 개장한다.

쌇O_CREAT
#엑세스 플래그

O_NDELAY UNIX 호환용, 여기서는 사용되지 않는다.
O_RDONLY 읽기 전용으로 개방
O_WRONLY 쓰기 전용으로 개방
O_RDWR 읽기와 쓰기용으로 개방
O_APPEND 이 플래그가 설정된 경우 파일 포인터가 파일끝으로 설정된다.
O_CREAT 이 플래그는 파일이 존재할 경우 영향이 없다. 파일이 존재 하지 않을 경우 파일이 만들어 지고 mode 비트가 파일의 속성을 결정하는데 사용된다
O_TRUNC 파일이 존재하는 경우 파일의 크기는 0 으로 되지만, 파일의 속성은 변경되지 않는다.
O_EXCL O_CREAT 와 같이 사용된다. 파일이 존재하는 경

우 에러가 리턴된다.
O_BINARY 파일을 이진 모드로 개방한다.
O_TEXT   파일을 텍스트 모드로 개장한다.

쌇O_EXCL
#엑세스 플래그

O_NDELAY UNIX 호환용, 여기서는 사용되지 않는다.
O_RDONLY 읽기 전용으로 개방
O_WRONLY 쓰기 전용으로 개방
O_RDWR   읽기와 쓰기용으로 개방
O_APPEND 이 플래그가 설정된 경우 파일 포인터가 파일끝
으로 설정된다.
O_CREAT   이 플래그는 파일이 존재할 경우 영향이 없다.
파일이 존재 하지 않을 경우 파일이 만들어 지고
mode 비트가 파일의 속성을 결정하는데 사용된다
O_TRUNC   파일이 존재하는 경우 파일의 크기는 0 으로 되
지만, 파일의 속성은 변경되지 않는다.
O_EXCL   O_CREAT 와 같이 사용된다. 파일이 존재하는 경
우 에러가 리턴된다.
O_BINARY 파일을 이진 모드로 개방한다.
O_TEXT   파일을 텍스트 모드로 개장한다.

쌇O_RDONLY
#엑세스 플래그

O_NDELAY UNIX 호환용, 여기서는 사용되지 않는다.
O_RDONLY 읽기 전용으로 개방
O_WRONLY 쓰기 전용으로 개방
O_RDWR   읽기와 쓰기용으로 개방
O_APPEND 이 플래그가 설정된 경우 파일 포인터가 파일끝
으로 설정된다.
O_CREAT   이 플래그는 파일이 존재할 경우 영향이 없다.
파일이 존재 하지 않을 경우 파일이 만들어 지고
mode 비트가 파일의 속성을 결정하는데 사용된다
O_TRUNC   파일이 존재하는 경우 파일의 크기는 0 으로 되
지만, 파일의 속성은 변경되지 않는다.
O_EXCL   O_CREAT 와 같이 사용된다. 파일이 존재하는 경
우 에러가 리턴된다.
O_BINARY 파일을 이진 모드로 개방한다.
O_TEXT   파일을 텍스트 모드로 개장한다.

쌿O_RDWR
#엑세스 플래그

O_NDELAY UNIX 호환용, 여기서는 사용되지 않는다.
O_RDONLY 읽기 전용으로 개방
O_WRONLY 쓰기 전용으로 개방
O_RDWR   읽기와 쓰기용으로 개방
O_APPEND 이 플래그가 설정된 경우 파일 포인터가 파일끝
          으로 설정된다.
O_CREAT   이 플래그는 파일이 존재할 경우 영향이 없다.
          파일이 존재 하지 않을 경우 파일이 만들어 지고
          mode 비트가 파일의 속성을 결정하는데 사용된다
O_TRUNC   파일이 존재하는 경우 파일의 크기는 0 으로 되
          지만, 파일의 속성은 변경되지 않는다.
O_EXCL   O_CREAT 와 같이 사용된다. 파일이 존재하는 경
          우 에러가 리턴된다.
O_BINARY 파일을 이진 모드로 개방한다.
O_TEXT   파일을 텍스트 모드로 개장한다.

쌿O_TEXT
#엑세스 플래그

O_NDELAY UNIX 호환용, 여기서는 사용되지 않는다.
O_RDONLY 읽기 전용으로 개방
O_WRONLY 쓰기 전용으로 개방
O_RDWR   읽기와 쓰기용으로 개방
O_APPEND 이 플래그가 설정된 경우 파일 포인터가 파일끝
          으로 설정된다.
O_CREAT   이 플래그는 파일이 존재할 경우 영향이 없다.
          파일이 존재 하지 않을 경우 파일이 만들어 지고
          mode 비트가 파일의 속성을 결정하는데 사용된다
O_TRUNC   파일이 존재하는 경우 파일의 크기는 0 으로 되
          지만, 파일의 속성은 변경되지 않는다.
O_EXCL   O_CREAT 와 같이 사용된다. 파일이 존재하는 경
          우 에러가 리턴된다.
O_BINARY 파일을 이진 모드로 개방한다.
O_TEXT   파일을 텍스트 모드로 개장한다.

쌿O_TRUNC
#엑세스 플래그

O_NDELAY UNIX 호환용, 여기서는 사용되지 않는다.
O_RDONLY 읽기 전용으로 개방
O_WRONLY 쓰기 전용으로 개방
O_RDWR   읽기와 쓰기용으로 개방
O_APPEND 이 플래그가 설정된 경우 파일 포인터가 파일끝
          으로 설정된다.
O_CREAT   이 플래그는 파일이 존재할 경우 영향이 없다.
          파일이 존재 하지 않을 경우 파일이 만들어 지고
          mode 비트가 파일의 속성을 결정하는데 사용된다
O_TRUNC   파일이 존재하는 경우 파일의 크기는 0 으로 되
          지만, 파일의 속성은 변경되지 않는다.
O_EXCL   O_CREAT 와 같이 사용된다. 파일이 존재하는 경
          우 에러가 리턴된다.
O_BINARY 파일을 이진 모드로 개방한다.
O_TEXT   파일을 텍스트 모드로 개장한다.

쌇O_WRONLY
#엑세스 플래그

O_NDELAY UNIX 호환용, 여기서는 사용되지 않는다.
O_RDONLY 읽기 전용으로 개방
O_WRONLY 쓰기 전용으로 개방
O_RDWR   읽기와 쓰기용으로 개방
O_APPEND 이 플래그가 설정된 경우 파일 포인터가 파일끝
          으로 설정된다.
O_CREAT   이 플래그는 파일이 존재할 경우 영향이 없다.
          파일이 존재 하지 않을 경우 파일이 만들어 지고
          mode 비트가 파일의 속성을 결정하는데 사용된다
O_TRUNC   파일이 존재하는 경우 파일의 크기는 0 으로 되
          지만, 파일의 속성은 변경되지 않는다.
O_EXCL   O_CREAT 와 같이 사용된다. 파일이 존재하는 경
          우 에러가 리턴된다.
O_BINARY 파일을 이진 모드로 개방한다.
O_TEXT   파일을 텍스트 모드로 개장한다.

쌇float.h
#헤더파일 FLOAT.H

함수

`_clear87` `_control87` `_fpreset`
`_status87`

쌿_clear87
#함수 _CLEAR87

기 능 부동 소숫점 상태 워드를 초기 상태로 한다.
구 문 unsigned int _clear87(void);
프로토타입 `float.h`
기능 설명 8087/80287의 상태 워드와 8087/80287예외 처리기에 의해 탐색괸 조건과의 조합인 부동 소숫점 상태 워드를 초기 상태로 한다.
리 턴 값 리턴되는 값인 비트가 초기 상태로 되기 전의 부동소숫전 상태를 나타낸다.
참 조 `_control87`,`_fpreset`,`_status87`

쌿_control87
#함수 _control87

기 능 부동 소숫점 제어 워드를 제어한다.
구 문 unsigned int _control87(unsigned int new, unsigned int mask);
프로토타입 `float.h`
기능 설명 현재의 부동소숫점 제어 워드를 검색하거나 변경한다. 부동 소숫점 제어 워드는 unsigned int 로서 한 비트 한 비트씩 어떤 특정 소숫점 패키지 (배정도, 무한대, 반올림 모드)로 지정한다.

이와 같은 모드를 변경함으로써 사용자가 부동 소숫점을 mask할 수 있다.
리 턴 값 새로운 부동소숫점 제어 워드의 값을 리턴한다. 즉 리턴된 값에 있는 비트는 부동 소숫점 제어 워드를 나타낸다.
참 조 `clear87`,`_fpreset`,`signal`,`status87`

쌿_fpreset
#함수 _FPRESET

기 능 부동 소숫점 연산 패키지를 다시 초기화한다.
구 문 void _fpreset(void);

프로토타입 `float.h`
기능 설명 부동 소숫점 math 패키지를 다시 초기화 한다
이 함수는 때때로`system`이나`exec`혹은
`spawn`함수 등과 같이 사용된다.

DOS 버전 2.0과 3.0 이전에서 8087/80287 Cop
rocessor를 프로그램에서 사용할 경우`system`
`exec`,`spawn`함수에 의해서 실행되는 생성
프로세스가 모 프로세스의 부동 소숫점 상태
를 변경하는 수가 있다. 그러므로 8087/80287
을 사용하는 경우 다음과 같은 사전 조치를
취해야 한다.

* 부동소숫점 계산이 수행되고 있는 동안에는
system,spawn,exec 을 수행하면 않된다.
* 생성 프로세스가 8087/80287로 부동 소숫점
연산을 수행하는 경우에는 system,spawn,ex
ec을 후에 _freset을 호출한다.

리 턴 값 없음
이 식 성 `_clear87`,`_control87`,`exec`,`spawn`,
`_status87`,`system

쌇_status87
#함수 _STATUS87

기 능 부동 소숫점 상태를 조사한다.
구 문 unsigned int _status87(void);
프로토타입 `float.h`
기능 설명 부동 소숫점 상태 워드를 갖는다. 이 상태 워
드는 8087/80287 상대단어와 8087/80287 예외
조정기에 의해 탐지되는 다른 조건과의 조합
이다.
리 턴 값 부동 소숫점의 상태를 리턴한다.
참 조 `_clear87`,`_control87`,`_fpreset`

쌇graphics.h
#헤더파일 GRAPHICS.H

함수

`arc` `bar` `bar3d`

`circle` `cleardevice` `clearviewport`
`closegraph` `detectgraph` `drawpoly`
`ellipse` `fillellipse` `fillpoly`
`floodfill` `getarccoords` `getbkcolor`
`getcolor` `getdefaultpalette` `getdrivername`
`getfillpattern` `getfillsettings` `getgraphmode`
`getimage` `getlinesettings` `getmaxcolor`
`getmaxmode` `getmaxx` `getmaxy`
`getmodename` `getmoderange` `getpalette`
`getpalettesize` `getpixel` `gettextsettings`
`getviewsettings` `getx` `gety`
`graphdefaults` `grapherrormsg` `_graphfreemem`
`_graphgetmem` `graphresult` `imagesize`
`initgraph` `installuserdriver` `installuserfont`
`line` `linerel` `lineto`
`moverel` `moveto` `outtext`
`outtextxy` `pieslice` `putimage`
`rectangle` `registerbgidriver` `registerbgifont`
`restorecrtmode` `sector` `setactivepage`
`setallpalette` `setaspectratio` `setbkcolor`
`setcolor` `setfillpattern` `setfillstyle`
`setgraphbufsize` `setgraphmode` `setlinestyle`
`setpalette` `setrgbpalette` `settextjustify`
`settextstyle` `setusercharsize` `setviewport`
`setvisualpage` `setwritemode` `texthight`
`textwidth`

쌇arc
#함수 ARC

기 능 원호(ARC)를 그린다.
구 문 #include <graphics.h>
void far arc(int x, int y,int stangle,
int endangle,int radius);
프로토타입 `graphics.h`
기능 설명 radius에 의해 주어진 반경으로 현재 작업하
고 있는 도형색의 중심점(x,y)에서 원호에를
그린다. arc는 stangle부터 endangle까지 이
다. stangle이 0이고 endangle이 360이면 arc
는 완전한 원을 그리게 된다.

arc 각은 3시에 0도, 12시에 90도 등 반시계 방향으로 계산한다.
linestyle 파라미터는 호, 원, 타원, 또는 부채꼴에 영향을 주지 않으며 단지 thickness파라미터만이 사용된다.

리 턴 값 없음

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `circle`,`ellipse`,`fillellipse`, `getarccoords`,`sector`

쌇bar

#함수 BAR

기 능 2차원의 bar를 그린다.

구 문 #include <graphics.h>
void far bar(int left,int top,
int right,int bottom);

프로토타입 `graphics.h`

기능 설명 bar 는 여백을 채운 직사각형의 2차원 의 막대모양을 그린다. bar는 현재의 fill 패턴과 fill 색상으로 채워진다. bar는 윤과선을 그리지 않으므로 윤과선을 그리기 위해서는 depth 가 0인 bar3d를 사용한다.

리 턴 값 없음

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `bar3d`,`rectangle`,`setcolor`, `setfillstyle`

쌇bar3d

#함수 BAR3D

기 능 3차원의 bar를 그린다.

구 문 #include <graphics.h>
void far bar3d(int left, int top,
int right, int bottom,
int depth, int topflag);

프로토타입 `graphics.h`

기능 설명 bar3d는 3차원의 직사각형 bar를 그리며, 현재의 fill 패턴과 색상을 사용하여 채운다.

3차원 bar의 윤곽선은 현색의 선의 색과 모양으로 그려진다. 픽셀 단위는 bar의 깊이는 depth 에 의해 주어진다.

topflag는 맨 위의 bar를 채울것 인지를 결정한다. topflag 에 0을 설정하면 윗면이 채워지지 않으며 0이외의 값을 설정하면 모든 면이 채워진다.
bar3d의 전형적인 depth를 계산하기 위하여 다음과 같이 bar 넓이의 25% 만을 취한다.

bar3d(left,top,right,bottom,(right-left)/4,1);

리 턴 값 없음
이 식 성 bar와 같음
참 조 `bar`,`rectangle`,`setcolor`,`setfillstyle`,`setlinestyle`

쌿circle
#함수 CIRCLE

기 능 중심(x,y) 에서 주어진 반지름으로 원을 그린다.
구 문 #include <graphics.h>
void far circle(int x,int y,int radius);
프로토타입 `graphics.h`
기능 설명 radius에서 지정된 반지름과 중심(x,y)을 갖는 현재의 묘사 색상으로 원을 그린다. 여기서 linestyle 은 호, 원, 타원, 그리고 부채꼴 등에 아무런 영양읗 끼치지 않으며 단지 thickness 파라미터 만이 사용된다.
리 턴 값 없음
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참 조 `arc`,`ellipse`,`fillellipse`,`getaspectratio`,`sector`,`setaspectratio`

쌿cleardevice
#함수 CLEARDEVICE

기 능 그래픽 화면을 초기화 한다.

구　　문　#include <graphics.h>
　　void far cleardevice(void);
프로토타입　`graphics.h`
기능 설명　그래픽 화면을 지우고 현재 위치(CP)를 (0,0)
　　으로 이동시킨다.
리 턴 값　없음
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘
　　IBM-PC 와 호환기종에 한한다.
참　　조　`clearviewport`

쌇clearviewport
#함수 CLEARVIEWPORT

기　　능　현재의 뷰포트를 최기화 시킨다.
구　　문　#include <graphics.h>
　　void far clearviewport(void);
프로토타입　`graphics.h`
기능 설명　현재의 뷰포트를 지우고 현재 위치(CP)를 좌
　　상귀로 이동 시킨다.
리 턴 값　없음
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘
　　IBM-PC 와 호환기종에 한한다.
참　　조　`cleardevice`,`getviewsettings`,
　　`setviewport`

쌇closegraph
#함수 CLOSEGRAPH

기　　능　그래픽 시스템을 종료한다.
구　　문　#include <grahpics.h>
　　void far closegraph(void);
프로토타입　`graphics.h`
기능 설명　그래픽 시스템 용으로 할당된 모든 메모리를
　　소거한 후 initgraph 를 호출하기 이전으로
　　돌아간다.
리 턴 값　없음
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘
　　IBM-PC 와 호환기종에 한한다.
참　　조　`initgraph`,`setgraphbufsize`

쌇detectgraph

#함수 DETECTGRAPH

기　　능　하드웨어를 검사하여 사용될 드라이버와 모드
　　를 검사한다.
구　　문　#include <dos.h>
　　void far detectgraph(int far *graphdriver,
　　int far *graphmode);
프로토타입　`graphics.h`
기능 설명　사용자 시스템의 그래픽 어댑터에 가장 적합
　　한 해상도를 제공하는 모드를 선택한다. 이
　　때 그래픽 하드웨어가 탐색되지 않으면 *gra
　　phdriver는 -2로 설정되고 graphresult도 -2
　　를 리턴하게 된다.

　　graphdriver는 정수로서 사용될 그래픽 드라
　　이버를 지정한다. 따라서 사용자는 graphics
　　.h 에 정의된 상수를 사용하여 *graphdriver
　　값을 부여할 수 있다.

　　*graphmode는 정수로서 초기의 그래픽 모드를
　　지정한다. (만일 *graphdriver 가 DETECT 가
　　아닌경우 *graphmode는 드라이버에 맞는 고해
　　상도로 설정한다.)
　　detectgraph를 호출하는 주된 이유는 initgra
　　ph를 불러내는 그래픽 모드를 취소하기 위해
　　서 이다.
리 턴 값　없음
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘
　　IBM-PC 와 호환기종에 한한다.
참　　조　`graphresult`,`initgraph`

쌇drawpoly
#함수 DRAWPLOY

기　　능　다각혀의 윤각을 그린다.
구　　문　#include <graphics.h>
　　void far drawpoly(int numpoints,
　　int dar *polypoints);
프로토타입　`graphics.h`
기능 설명　현재의 라인 형태와 색상을 사용해서 numpoin
　　ts 점을 연결하여 다각형을 그린다. polypoin

ts 는 연속된 정수 (numpoints X 2)를 가리키며 이들 정수의 각 쌍은 다각형 상의 x와 y좌표의 점을 나타낸다.

여기서 n개의 꼭지점을 갖는 패쇄형으로 그리기 위해서는 n+1개의 좌표를 drawpoly 에게 넘겨 주어야만 n개의 좌표가 0번째의 좌표와 같이 연결된다.

리 턴 값 다각형이 그려지고 있는 동안에 에러가 발생하면 graphresult 는 -6을 리턴한다.

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `fillpoly`,`floodfill`,`graphresult`, `setwritemode`

쌿ellipse

#함수 ELLIPSE

기 능 타원의 원호를 그린다.

구 문 #include <graphics.h>
void far ellipse(int x,int y,int stangle, int endangle,int xradius,int yradius);

프로토타입 `graphics.h`

기능 설명 중심이 (x,y)이고 수편및 수직축이 각각 xradius, yradius 인 타원호를 현재 지정한 색상으로 그린다. 중심각이 stangle 에서 endangl 까지인 타원호가 그려진다.

완전한 타원을 그릴 경우에는 stangle=0 endangle=360 으로 설정한다. ellipse 의 각도는 반 시계 방향으로 3시 방향이 0도 12시 90도 등으로 설정된다. 여기서 linestyle 파라미터는 반원, 원, 타원, 그리고 부채꼴 등에 어떠한 영향도 미치지 않는다. 따라서 thickness 파라미터 만이 사용된다.

리 턴 값 없음

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `arc`,`circle`,`fillellipse`, `setaspectratio`,`sector`,`setaspectratio`

쌿fillellipse
#함수 FILLELLIPSE

기　　능　타원을 그린 후 칠한다.
구　　문　#include <graphics.h>
　　void far fillellipse(int x,int y,
　　int xradius,int yradius);
프로토타입　`graphics.h`
기능 설명　중심이 (x,y) 수평 및 수직축이 각각 xradius
　　,yradius로 주어진 타원을 현재 주어진 색상
　　과 현태로 그린다.
리 턴 값　없음
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘
　　IBM-PC 와 호환기종에 한한다.
참　　조　`arc`,`circle`,`ellipse`,`getaspectratio`,
　　`pieslice`,`setaspectratio`

쌿fillpoly
#함수 FILLPOLY

기　　능　다각형을 그린 후 칠한다.
구　　문　include <graphics.h>
　　void far fillpoly(int numpoints,
　　int far *polypoints);
프로토타입　`graphics.h`
기능 설명　현재 선의형태 및 색으로 이루어진 numpoints
　　개의 점을 사용하여 다각형의 윤곽을 그린다
　　음 현재 지정된 색상과 형태로 다각형을 그린
　　다.
　　polypoint는 일련의 (numpoints X 2) 정수를
　　가르킨다. 각 정수의 쌍은 다각형 상의 (x,y)
　　좌 표를 나타낸다.
리 턴 값　없음
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘
　　IBM-PC 와 호환기종에 한한다.
참　　조　`drawpoly`,`floddfill`,`graphresult`,
　　`setfillstyle`

쌿floodfill
#함수 FLOODFILL

기 능 경계 영역 내부를 칠한다.
구 문 #include <graphics.h>
void far floodfill(int x,int y,
int border);
프로토타입 `graphics.h`
기능 설명 비트맵 디바이스상에 경계지어진 내부 영역을 채운다. (x,y)를 포함하는 패쇄된 영역은 현재의 fill 형태와 색상으로 채워지고, 색상 border로 그어진 직선,곡선 만이 경계로 인식된다. 점(x,y)이 경계 영역 내부에 있으면 그 내부를 채우고 경계 밖에 있으면 경계 외부가 채워지게 된다.

다른 드라이버 와의 호환을 위해서 floodfill 보다는 fillpoly를 사용하는것이 좋다.
IBM-8514 드라이버에서는 floodfill 이 작동하지 않는다.
리 턴 값 영역을 채우는 도중 에러가 발생하면 graphresult 는 -7을 리턴한다.
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참 조 `drawpoly`,`fillpoly`,`graphresult`, `setcolor`,`setfillstyle`

쌇getarccoords
#함수 GETARCCOORDS

기 능 마지막으로 호출된 호(ARC)의 좌표축을 구한다.
구 문 #include <graphics.h>
void far getarccoords(struct arccoordtype far *arccords);
프로토타입 `graphics.h`
기능 설명 마지막으로 호출된 호(ARC)에 관한 정보를 가리키는 구조체 arccoordstype에 대입한다. 이 구조체의 요소들은 호의 중심점(x,y), 호의 시작점(xstart,ystart) 그리고 호의 종료지점(xend,yend) 등을 지정하는데 사용된다. 이 값은 호의 종료지점과 만나는 선을 그리는데 유용하다.
리 턴 값 없음

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘
          IBM-PC 와 호환기종에 한한다.
참 조 `arc`,`fillellipse`,`sector`

쌇getbkcolor
#함수 GETBKCOLOR

기 능 현재의 배경색을 리턴한다.
구 문 #include <graphics.h>
          int far getbkcolor(void);
프로토타입 `graphics.h`
기능 설명 현재의 배경색을 리턴한다. (자세한 사항은 s
          etbkcolor 의 표를 참조하기 바란다.)
리 턴 값 현재 화면의 배경색을 리턴한다.
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘
          IBM-PC 와 호환기종에 한한다.
참 조 `getcolor`,`getmaxcolor`,`getpalette`,
          `setbkcolor`

쌇getcolor
#함수 GETCOLOR

기 능 그리고 있는 색을 리턴한다.
구 문 #include <graphics.h>
          int far setcolor(void);
프로토타입 `graphics.h`
기능 설명 그리고 있는 색을 리턴한다. 그리는 색은 선
          이 그려질 때 픽셀에 설정되는 값이다.
리 턴 값 getcolor 는 그리고 있는 색을 리턴한다.
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘
          IBM-PC 와 호환기종에 한한다.
참 조 `getbkcolor`,`getmaxcolor`,`getpalette`,
          `setcolor`

쌇getdefaultpalette
#함수 GETDEFAULTPALETTE

기 능 팔레트 정의 구조체를 리턴한다.
구 문 #include <graphics.h>
          void far *getdefaultpalette(void);
프로토타입 `graphics.h`

기능 설명 initgraph중에 드라이버에 의해 초기화된 파레트를 포함하고 있는 palettetype 구조를 갖는다.
리 턴 값 현재 드라이버가 초기화 되었을 때 이 드라이버에 의해 설정된 디폴트 파레트에 대한 포인터를 리턴한다.
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참 조 `getpalette`,`initgraph`

쌿getdrivername
#함수 GETDRIVERNAME

기 능 현재 로드되어 있는 그래픽 드라이버의 이름을 갖는 문자열의 포인터를 리턴한다.
구 문 #include <graphics.h>
char far *getdrivername(void);
프로토타입 `graphics.h`
기능 설명 initgraph를 호출한 후에 getdrivername은 현재 로드되어 있는 드라이버의 이름을 리턴한다.
리 턴 값 현재 로드되어 있는 드라이버의 이름을 리턴한다.
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참 조 `initgraph`

쌿getfillpattern
#함수 GETFILLPATTERN

기 능 사용자가 정의한 fill-pattern을 메모리에 복사한다.
구 문 #include <graphics.h>
void far getfillpattern(char far *pattern);
프로토타입 `graphics.h`
기능 설명 setfillpattern에 따라 사용자가 정의한 fill-pattern을 pattern이 지정한 영역에 복사한다. 이 영역은 8바이트가 필요하다. pattern은 연속한 8바이트를 가리키는 포인터이며 여기서 각 바이트는 그 패턴의 8개의 픽셀에 대응 된다. 한 개의 패턴 바이트에서 1비트가 1

로 대응될 때마다 대응되는 패턴이 플로트 된다.

리 턴 값 없음

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `setfillsettings`,`setfillpattern`

쌇getfillsettings

#함수 GETFILLSETTINGS

기 능 정의된 fill-pattern 과 색상에 대한 정보를 갖는다.

구 문 #include <graphics.h>
void far getfillsettings(
struct fillsettingstype far *fillinfo);

프로토타입 `graphics.h`

기능 설명 fillinfo가 지정한 fillsettingstype 구조체에 현재의 fill형태와 색상에 관한 정보를 채워 넣는다.

리 턴 값 없음

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `getfillpattern`,`setfillpattern`, `setfillstyle`

쌇getgraphmode

#함수 GETGRAPHMODE

기 능 현재 그래픽 모드를 리턴한다.

구 문 #include <graphics.h>
int far getgraphmode(void);

프로토타입 `graphics.h`

기능 설명 getgraphmode 를 호출하기 전에 initgraph 를 호출한다. graphics.h 에서 정의된 graphics_mode는 미리 정의된 그래픽 모드에 대해서 이름을 부여한다. 이와 같이 열거한 값들을 표로 작성하여 나타낸것이 있는데 initgraph를 참고하기 바란다.

리 턴 값 initgraph나 segraphmode에 의해 설정된 그래픽 모드를 리턴한다.

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘

IBM-PC 와 호환기종에 한한다.
참 조 `getmoderange`,`restorecrtmode`,
`setgraphmode`

쌇getimage
#함수 GETIMAGE

기 능 지정된 지역의 비트 이미지를 메모리에 저장
한다.
구 문 #include <graphics.h>
void far getimage(int left,int top,
int right,int bottom,void far *bitmap);
프로토타입 `graphics.h`
기능 설명 화면의 이미지를 메모리에 복사한다. 화면상
의 left,top,right,bottom 을 대각으로 하는
장방향 영역의 비트 이미지를 bitmap 에서
지정한 곳으로 복사한다. 이 영역 첫번째 두
워드는 가로와 세로의 크기를 갖고 나머지는
이미지 자체가 된다.
리 턴 값 없음
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘
IBM-PC 와 호환기종에 한한다.
참 조 `imagesize`,`putimage`,`putpixel`

쌇getlinesettings
#함수 GETLINESETTINGS

기 능 현재 선의 스타일, 패턴, 굵기를 갖는다.
구 문 #include <graphics.h>
void far getlinesettings(
struct linesettings far *lineinfo);
프로토타입 `graphics.h`
기능 설명 지정한 서의 스타일과 패턴, 굵기에 관한 정
보를 lineinfo가 지정하는 linesettings 구조
체에 채워 넣는다. linestyle 은 선이 그려지
는 스타일을 실선, 점선, 일점 쇄선, 파선,
등으로 지정한다. graphics.h 에 정의되어 있
는 line_style 은 다음과 같다.

이름	값	내용

SOLID_LINE	0	실선
DOTTED_LINE	1	점선
CENTER_LINE	2	일점 쇄선
DASHED_LINE	3	파선
USERBIT_LINE	4	사용자 선

thickness 는 선의 굵기를 지정하며 다음과 같이 지정된다.

이름	값	내용
NORM_WIDTH	1	1 픽셀
RHICK_WIDTH	3	3 픽셀

리 턴 값 없음
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참 조 `setlinestyle`

쌿getmaxcolor
#함수 GETMAXCOLOR

기 능 setcolor 함수에 전달될 수 있는 최대 색상값을 리턴한다.
구 문 #include <graphics.h>
int far getmaxcolor(void);
프로토타입 `getmaxcolor`
기능 설명 setcolor에 전달될 수 있는 현재의 그래픽 드라이버와 모드에 대해서 최대로 가능한 색상값을 리턴한다. 예를 들어 EGA에서는 15를 리턴한다. 이는 0-15 까지 16색을 쓸 수 있음을 의미한다.
리 턴 값 사용할 수 있는 최대의 색상값을 리턴한다.
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참 조 `getbkcolor`,`getcolor`,`getpalette`, `getpalettesize`,`setcolor`

쌿getmaxmode
#함수 GETMAXMODE

기 능 현재의 드라이버 에서 사용할 수 있는 그래픽
의 최대 모드를 구한다.
구 문 #include <graphics.h>
int far getmaxmode(void);
프로토타입 `graphics.h`
기능 설명 사용자가 현재 로드된 드라이버에 대한 최대
모드 번호를 찾을 수 있도록 해준다. 여기서
최소 모드는 0이된다.
리 턴 값 현재의 드라이버 에서 사용할 수 있는 그래픽
의 최대 모드를 구한다.
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘
IBM-PC 와 호환기종에 한한다.
참 조 `getmodename`,`getmoderange`

쌿getmaxx
#함수 GETMAXX

기 능 화면상 x 좌표의 최대값을 구한다.
구 문 #include <graphics.h>
int far getmaxx(void);
프로토타입 `graphics.h`
기능 설명 현재의 그래픽 드라이버와 모드에서 화면상의
최대 x 값을 리턴한다. 예를들어 640x480모드
에서는 639가 리턴된다. 이 함수는 화면상에
서 영역의 경계와 중심을 설정하는데 중요한
역활을 한다.
리 턴 값 화면상 x 좌표의 최대 값을 리턴한다.
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘
IBM-PC 와 호환기종에 한한다.
참 조 `getmaxy`,`getx`

쌿getmaxy
#함수 GETMAXY

기 능 화면상 y 좌표의 최대값을 구한다.
구 문 #include <graphics.h>
int far getmaxy(void);
프로토타입 `graphics.h`
기능 설명 현재의 그래픽 드라이버와 모드에서 화면상의
최대 y 값을 리턴한다. 예를들어 640x480모드

에서는 479가 리턴된다. 이 함수는 화면상에서 영역의 경계와 중심을 설정하는데 중요한 역활을 한다.

리 턴 값 화면상 y 좌표의 최대 값을 리턴한다.

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `getmaxx`,`gety`

쌇getmodename

#함수 GETMODENAME

기 능 지정된 그래픽 모드의 이름을 갖고 있는 문자열 포인터를 리턴한다.

구 문 #include <graphics.h>
char far *getmodename(int mode_number);

프로토타입 `graphics.h`

기능 설명 그래픽 모드 변환을 입력받아서 이에 대응되는 그래픽 모드의 이름을 갖고 있는 문자열을 리턴한다. 모드의 각 이름은 각 드라이버에 삽입되어 있다.

리 턴 값 그래픽 모드의 이름을 갖고 있는 문자열의 포인터를 리턴한다.

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `getmaxmode`,`getmoderange`

쌇getmoderange

#함수 GETMODERANGE

기 능 주어진 그래픽 드라이버에 대한 모드의 범위를 구한다.

구 문 #include <graphics.h>
void far getmoderange(int graphdriver,
int far *lomode,int far *himode);

프로토타입 `graphics.h`

기능 설명 주어진 graphdriver에 대해 유효한 그래픽 모드의 범위를 갖는다. 가장 낮은값은 *lomode에 리턴되고 가장 높은값은 *himode에 리턴된다. graphdriver가 부적당한 그래픽 드라이버를 지정하는 경우 *lomode와 *himode는 -1로

설정된다. graphdriver의 값이 -1인 경우 현재 사용하고 있는 그래픽 드라이버가 된다.

리 턴 값 없음

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `getgraphmode`,`getmaxmode`,`getmodename`, `initgraph`,`setgraphmode`

쌇getpalette

#함수 GETPALETTE

기 능 현재 팔레트에 대한 정보를 구한다.

구 문 #include <graphics.h>
void far getpalette(struct palettetype far *palette);

프로토타입 `graphics.h`

기능 설명 palette가 가리키는 palettetype 구조체에 현재 팔레트의 크기와 색상에 관한 정보를 채워 넣는다. getpalette에 의해 사용되는 MAXCOLOR 상수와 palettetype 구조체는 graphics.h에서 다음과 같이 정의된다.

```
#define MAXCOLOR 15
struct palettetype {
    unsigned char size;
    unsigned char colors[MAXCOLOR + 1];
};
```

size는 지정모드에서 지정된 그래픽 드라이버에 대한 팔레트 색상의 수를 제공한다.
colors는 팔레트 안의 각 요소에 대한 실제 색상 수를 갖는 size 바이트의 배열이다.

리 턴 값 없음

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `getbkcolor`,`getcolor`,`getdefaultpalette`, `getmaxcolor`,`setallpalette`,`setpalette`

쌇getpalettesize

#함수 GETPALETTESIZE

기 능 팔레트 색상표의 크기를 리턴한다.

구　　문　#include <graphics.h>
　　int far getpalettesize(void);
프로토타입　`graphics.h`
기능 설명　현재 그래픽 모드에 대해 얼마나 많은 팔레트 요소가 설정될 수 있는가를 결정하는 데 사용된다. 예를 들면 색상 모드에서 EGA는 16을 리턴 하게 된다.
리 턴 값　현재 팔레트에서 팔레트 요소의 수를 리턴한다.
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참　　조　`setpalette`,`setallpalette`

쌿getpixel
#함수 GETPIXEL

기　　능　지정된 픽셀의 색상을 구한다.
구　　문　#include <graphics.h>
　　unsigned far getpixel(int x,int y);
프로토타입　`graphics.h`
기능 설명　좌표(x,y)에 위치한 픽셀의 색상을 갖는다.
리 턴 값　주어진 픽셀의 색상을 리턴한다.
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참　　조　`getimage`,`putpixel`

쌿gettextsettings
#함수 GETTEXTSETTINGS

기　　능　현재 그래픽 텍스트 폰트에 관한 정보를 구한다.
구　　문　#include <graphics.h>
　　void far gettextsettings(struct
　　textsettingstype far *texttypeinfo);
프로토타입　`graphics.h`
기능 설명　textinfo 가 지정한 textsettingstype 구조체에 텍스트 폰트, 방향, 크기, 정렬(justification)에 관한 정보를 기입한다.
리 턴 값　없음
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참　　조　`outtext`,`outtextxy`,`settextjustify`,
　　`settextstyle`,`setusercharsize`,
　　`texthight`,`textwidth`

쌇getviewsettings
#함수 GETVIEWSETTINGS

기　　능　현재 뷰포트에 관한 정보를 구한다.
구　　문　#include <graphics.h>
　　void far getviewsettings(
　　struct viewporttype *viewport);
프로토타입　`graphics.h`
기능 설명　viewport가 지정한 viewporttype 구조체에 현
　　재 뷰포트에 관한 정보를 채워 넣는다.
리 턴 값　없음
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘
　　IBM-PC 와 호환기종에 한한다.
참　　조　`clearviewport`,`getx`,`gety`,`setviewport`

쌇getx
#함수 GETX

기　　능　그래픽의 x 좌표 위치를 리턴한다.
구　　문　#include <graphics.h>
　　int far getx(void);
프로토타입　`graphics.h`
기능 설명　그래픽의 x좌표 위치를 리턴한다. 그 값은 뷰
　　포트 내에서 상대적인 위치가 된다.
리 턴 값　현재 위치의 x 좌표를 리턴한다.
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘
　　IBM-PC 와 호환기종에 한한다.
참　　조　`getmaxx`,`getmaxy`,`getviewsettings`,`gety`

쌇gety
#함수 GETY

기　　능　그래픽의 y 좌표를 리턴한다.
구　　문　#include <graphics.h>
　　int far gety(void);
프로토타입　`graphics.h`
기능 설명　그래픽의 y좌표 위치를 리턴한다. 그 값은 뷰

포트 내에서 상대적인 위치가 된다.
리 턴 값 현재 위치의 y 좌표를 리턴한다.
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참 조 `getmaxx`,`getmaxy`,`getviewsettings`,`getx`

쌇graphdefaults
#함수 GRAGHDEFAULTS

기 능 모든 그래픽 설정을 디폴트로 다시 설정한다.
구 문 #include <graphics.h>
void far graphdefaults(void);
프로토타입 `graphics.h`
기능 설명 모든 그래픽 설정을 디폴트로 다시 지정한다.

* 뷰포트를 전체 화면으로 설정한다.
* 현재 위치를 (0,0)으로 이동시킨다.
* 디폴트 팔레트 색상, 배경색, 드로잉 색상을 설정한다.
* 디폴트 fill 스타일과 패턴을 설정한다.
* 디폴트 텍스트 폰트와 정렬(justification)을 설정한다.

리 턴 값 없음
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참 조 `initgraph`

쌇grapherrormsg
#함수 GRAPHERRORMSG

기 능 에러 메시지 문자열을 리턴한다.
구 문 #include <graphics.h>
char far *grapherrormsg(int errorcode);
프로토타입 `graphics.h`
기능 설명 graphresult에 의해 리턴된 값인 errorcode와 연관된 에러 메시지 문자와 포인터를 리턴한다.
리 턴 값 에러 메시지 문자열에 대한 포인터를 리턴한다.
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `graphresult`

쌇_graphfreemem
#함수 _GRAPHFREEMEM

기 능 사용자가 그래픽 메모리 해제를 다시 정의한다.
구 문 #include <graphics.h>
void far _graphfreemem(void far *ptr,
unsigned size);
프로토타입 `graphics.h`
기능 설명 _graphgetmem을 통해 할당된 메모리를 해제하기 위해서 그래픽 라이브러리는 _graphfreemem을 호출한다. 사용자 버전인 _graphfreemem을 정의함으로써 그래픽 라이브러리의 메모리 운영을 통제할 수 있다. 이 루틴의 디폴트 버전은 단지 free를 호출한다. 이 함수는 사용자가 다시 정의할 수 있는 함수로 프로그램중에 정의하면 이것이 우선적으로 사용된다.
리 턴 값 없음
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참 조 `_graphfgetmem`,`setgraphbufsize`

쌇_graphgetmem
#함수 _GRAPHGETMEM

기 능 사용자가 그래픽 메모리 할당을 재정의 한다. 그래픽 메모리를 할당한다.
구 문 #include <graphics.h>
void far *_graphgetmem(unsigned size);
프로토타입 `graphics.h`
기능 설명 내부 버퍼, 그래픽 드라이버, 그리고 문자체 폰트를 위한 메모리를 할당하기 위해서 그래픽 라이브러리에 있는 루틴은 _graphgetmem을 호출한다.
이 루틴의 디폴트 버전은 단지`malloc`만을 호출한다. 이 함수는 사용자 재정의가 가능한 함수로 재정의가 되면 그것이 우선적으로 사용 된다.
리 턴 값 없음

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘
IBM-PC 와 호환기종에 한한다.
참 조 `graphfreemem`,`initgraph`,`setgraphbufsize`

쌇graphresult
#함수 GRAPHRESULT

기 능 최후에 발생한 그래픽 작동에 대한 에러 코드
를 구한다.
구 문 #include <graphics.h>
int far graphresult(void);
프로토타입 `graphics.h`
기능 설명 가장 최근에 발생한 그래픽 작동의 에러에 대
한 에러 코드를 리턴한다. graphresult에 의
해 유지되는 변수는 호출된 후에 다시 0으로
설정된다. 그러므로 사용자는 graphresult의
값을 임시변수에 담아서 사용하는것이 좋다.
리 턴 값 없음
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘
IBM-PC 와 호환기종에 한한다.
참 조 `detectgraph`,`drawpoly`,`fillpoly`,
`floodfill`,`grapherrmsg`,`initgraph`,
`pieslice`,`registerbgidriver`,
`registerbgifont`,`setallpalette`,
`setcolor`,`setfillstyle`,`setgraphmode`,
`setlinestyle`,`setpalette`,
`settextjustify`,`settextstyle`,
`setusercharsize`,`setviewport`,
`setvisualpage`

쌇imagesize
#함수 IMAGESIZE

기 능 비트 이미지를 저장하는데 필요한 바이트 수
를 리턴한다.
구 문 #include <graphics.h>
unsigned far imagesize(int left,int top,
int right,int bottom);
프로토타입 `graphics.h`
기능 설명 비트 이미지를 저장하는 데 필요한 메모리 용
량을 리턴한다. 선택된 이미지의 크기가 64KB

-1바이트 보다 크거나 같으면 imagesize는 (0xffff)-1을 리턴한다.

리 턴 값 필요로 하는 메모리의 크기를 바이트로 리턴한다.

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `getimage`,`putimage`

쌇initgraph

#함수 INITGRAPH

기 능 그래픽 시스템을 초기화 한다.

구 문 #include <graphics.h>

```
void far initgraph(int far *graphdriver,
                   int far *graphmode,
                   char far *pathtodriver);
```

프로토타입 `graphics.h`

기능 설명 디스크로부터 그래픽 드라이버를 로드하고 시스템을 그래픽 모드로 함으로써 그래픽 시스템을 초기화 한다.

initgraph는 그래픽 드라이버를 로드하고 시스템을 그래픽 모드로 한다. 사용자는 initgraph가 특정한 그래픽 드라이버와 모드를 사용하도록 명령하거나 또는 실행시에 부착된 비디오 어댑터를 자동 검사해서 해당되는 드라이버를 선택하도록 할 수 있다.

보통 initgraph는 _graphgetmem을 통해서 드라이버를 위한 메모리를 할당한 후 디스크로부터 적당한 .BGI 파일을 로드함으로써 그래픽 드라이버를 로드한다.

리 턴 값 항상 내부 에러코드를 설정한다. 에러가 없으면 코드를 0으로 설정한다. 에러발생시 *graphdriver는 -2,-3,-4또는 -5로 설정되고 graphresult도 같은값을 리턴한는데 다음과 같다.

-2 드래픽 카드를 찾을 수 없다.
-3 드라이버 파일을 찾을 수 없다.
-4 드라이버가 틀리다.

-5 드라이버를 로드할 메모리가 부족하다.

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `closegraph`,`detectgraph`,`getdrivername`, `getmoderange`,`graphdefaults`, `_graphgetmem`,`graphresult`, `installuserdriver`,`registerbgidriver`, `registerbgifont`,`restorecrtmode`, `setgraphbufsize`,`setgraphmode`

쌿installuserdriver
#함수 INSTALLUSERDRIVER

기 능 외부 지원 디바이스 드라이버를 BGI 디바이스 드라이버표에 등록한다.

구 문 #include <graphics.h>
int far installuserdriver(char far *name, int huge(*detect)(void));

프로토타입 `graphics.h`

기능 설명 외부에서 지원되는 디바이스 드라이버를 BGI 내부표에 추가 등록하는 함수이다. 파라미터 name은 새로운 디바이스 드라이버 파일(BGI) 명이고, 파라미터 detect는 새로운 드라이버의 자동검색 함수가 부속되어 있는 경우에 이를 가리키는 포인터가 된다. 이 자동검색 함수는 파라미터를 갖지 않으며 정수를 리턴한다.

리 턴 값 installuserdriver에 의해 리턴된 값은 새롭게 설치된 드라이버를 선택 하기 위해서 사용자가 initgraph에 전달하는 드라이버 번호 파라미터 이다.

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `initgraph`,`registerbgidriver`

쌿installuserfont
#함수 INSTALLUSERFONT

기 능 BGI 시스템에 내장되어 있지 않은 폰트 파일(.CHR)을 로드한다.

구　　문　#include <graphics.h>
　　int far installuserfont(char far *name);
프로토타입　`graphics.h`
기능 설명　name은 폰트를 가지고 있는 폰트 파일에 대한 경로명이 된다. 동시에 로드될 수 있는 폰트는 20가지 이다.
리 턴 값　대응되는 폰트를 선택하기 위해서 settextstyle에 전달되는 폰트 ID번호를 리턴한다.
　　내부 폰트 테이블이 모두 찾을 때는 -11이 리턴 된다.
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참　　조　`settextstyle`

쌇line
#함수 LINE

기　　능　두 점간에 직선을 긋는다.
구　　문　#include <graphics.h>
　　void far line(int x1,int y1,int x2,int y2);
프로토타입　`graphics.h`
기능 설명　현재의 위치를 업데이트하지 않은채 특정한 두점 (x1,y1)-(x2,y2)간을 지정된 형태와 굵기 그리고 색상으로 선을 그린다.
리 턴 값　없음
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참　　조　`linerel`,`lineto`,`setcolor`,
　　`setlinestyle`,`setwritemode`

쌇linerel
#함수 LINEREL

기　　능　현재의 위치(CP)로부터 적정 거리만큼 이동해서 선을 그린다.
구　　문　#include <graphics.h>
　　void far linerel(int dx,int dy);
프로토타입　`graphics.h`
기능 설명　CP에서 적정 거리에 있는 (dx,dy) 점 까지 선을 그린다. CP는 (dx,dy) 만큼 이동한다.
리 턴 값　없음

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘
IBM-PC 와 호환기종에 한한다.
참 조 `line`,`lineto`,`setcolor`,`setlinestyle`,
`setwritemode`

쌇lineto
#함수 LINETO

기 능 현재 위치(CP)에서부터 (x,y)까지 선을 그린
다.
구 문 #include <graphics.h>
void far lineto(int x,int y);
프로토타입 `graphics.h`
기능 설명 CP로 부터 (x,y)까지 선을 그리며 현재의 위
치(CP)는 (x,y)로 이동한다.
리 턴 값 없음
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘
IBM-PC 와 호환기종에 한한다.
참 조 `line`,`linerel`,`setcolor`,`setlinestyle`,
`setwritemode`

쌇moverel
#함수 MOVEREL

기 능 현재 위치(CP)를 적정 거리만큼 이동시킨다.
구 문 #include <graphics.h>
void far moverel(int dx,int dy);
프로토타입 `graphics.h`
기능 설명 현재의 위치(CP)를 x축으로 dx 픽셀 y축으로
dy 픽셀만큼 이동시킨다.
리 턴 값 없음
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘
IBM-PC 와 호환기종에 한한다.
참 조 `moveto`

쌇moveto
#함수 MOVETO

기 능 현재의 위치(CP)를 (x,y) 좌표로 이동한다.
구 문 #include <graphics.h>
void far moveto(int x,int y);

프로토타입 `graphics.h`
기능 설명 현재 위치(CP)를 뷰포트 위치(x,y)로 이동한
다.
리 턴 값 없음
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘
IBM-PC 와 호환기종에 한한다.
참 조 `moverel`

쌇outtext
#함수 OUTTEXT

기 능 뷰포트에서 문자열을 표시한다.
구 문 #include <graphics.h>
void far outtext(char far *textstring);
프로토타입 `graphics.h`
기능 설명 뷰포트에서 텍스트 문자열을 표시한다. 이 때
현재 설정된 정렬과 폰트, 방향, 크기를 사용
한다. outtext는 현재 위치(CP)에 textstring
을 출력한다. 수평의 텍스트 정렬이 LEFT_TEX
T이고 텍스트 정렬이 HORIZ_DIR로 설정되면 C
P의 x좌표는 textwidth(textstring)만큼 증가
한다.

그 외의 경우에는 CP는 변경되지 않고 그대로
남는다. 여러개의 폰트를 사용할 때 코드 호
환성을 유지하기 위해서는 textwidth와 texth
ight를 사용해서 문자열의 크기를 지정한다.
만일 문자열이 outtext를 사용해서 디폴트 폰
트로 인쇄될 경우, 현재의 뷰포트 밖에 있는
문자열은 잘려지게 됨을 주의 해야 한다.
또한 outtext는 그래픽 모드에서만 사용하여
야만 하며 텍스트 모드에서는 사용할 수 없다
리 턴 값 없음
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘
IBM-PC 와 호환기종에 한한다.
참 조 `gettextsettings`,`outtextxy`,
`settextjustify`,`texthight`,`textwidth`

쌇outtextxy
#함수 OUTTEXTXY

기 능 지정된 위치에 문자열을 표시한다.
구 문 #include <graphics.h>
void far outtextxy(int x, int y,
char far *textstring);
프로토타입 `graphics.h`
기능 설명 주어진 위치(x,y)에 있는 뷰포트 상에서 텍스트 문자열을 표시하는데, 이 때 정렬 선정과 폰트,방향 그리고 폰트를 사용한다.

여러개의 폰트를 사용할 때 코드 호환성을 유지하기 위해서는 textwidth 와 texthigght 를 사용해서 문자열의 크기를 지정한다.

만일 문자열이 outtext를 사용해서 디폴트 폰트로 인쇄될 경우, 현재의 뷰포트 밖에 있는 문자열은 잘려지게 됨을 주의 해야 한다.
또한 outtext는 그래픽 모드에서만 사용하여야만 하며 텍스트 모드에서는 사용할 수 없다
리 턴 값 없음
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참 조 `gettextsettings`,`outtext`,
`settextjustify`,`texthight`,`textwidth`

쌿pieslice
#함수 PIESLICE

기 능 내부가 색칠된 부채꼴을 그린다.
구 문 #include <graphics.h>
void far pieslice(int x, int y,
int stangle,int endangle,int radius);
프로토타입 `graphics.h`
기능 설명 반경이 radius이고 중심이 (x,y)인 내부가 칠해진 부채꼴을 그린다. 부채꼴은 stangle부터 endangle까지 그려진다. 이 부채꼴은 현재 지정된 색상으로 경계가 구분되고, fill 패턴과 fill 색상을 사용하여 채색된다.
리 턴 값 없음
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참 조 `fillellipse`,`graphresult`,`sector`,

`setfillstyle`

쌇putimage
#함수 PUTIMAGE

기　　능　비트 이미지를 화면에 출력한다.
구　　문　#include <graphics.h>
　　void far putimage(int left,int top,
　　void far *bitmap,int op);
프로토타입　`graphics.h`
기능 설명　이전에 getimage로 저장된 비트이미지를 화면
　　상에 출력하는데 그 이미지의 좌상귀는 (left
　　,top)이 된다. bitmap은 본래의 이미지가 저
　　장된 메모리 상의 영역을 가리킨다.

　　putimage 의 op 인수는 연산 조합을 지정하여
　　이미 화면에 있는 픽셀과 대응하여 서로 다른
　　결과를 나타낸다.

이름	값	내용
COPY_PUT	0	복사
XOR_PUT	1	XOR(exclusive or)
OR_PUT	2	OR(inclusive or)
AND_PUT	3	AND
NOT_PUT	4	역상

리 턴 값　없음
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘
　　IBM-PC 와 호환기종에 한한다.
참　　조　`getimage`,`imagesize`,`putpixel`,
　　`setvisualpage`

쌇putpixel
#함수 PUTPIXEL

기　　능　지정된 위치에 픽셀을 표시한다.
구　　문　#include <graphics.h>
　　void far putpixel(int x,int y,int color);
프로토타입　`graphics.h`
기능 설명　좌표(x,y)에 color가 지정한 색으로 점을 찍

는다.
리 턴 값 없음
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참 조 `getpixel`,`putimage`

쌇rectangle
#함수 RECTANGLE

기 능 직사각형을 그린다.
구 문 #include <graphics.h>
void far rectangle(int left,int top,
int right,int bottom);
프로토타입 `graphics.h`
기능 설명 지정된 선의 굵기, 형태 및 색으로 직사각형을 그린다. (left,top)이 좌상귀의 좌표 (right,bottom)은 우하귀의 좌표가 된다.
리 턴 값 없음
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참 조 `bar`,`bar3d`,`setcolor`,`setlinestyle`

쌇registerbgidriver
#함수 REGISTERBGIDRIVER

기 능 사용자 로드(user_loaded) 또는 링크-인(link-in)된 그래픽 드라이버 코드를 시스템에 등록한다.
구 문 #include <graphics.h>
int registerbgidriver(void *(*driver)
(void));
프로토타입 `graphics.h`
기능 설명 사용자로 하여금 드라이버 파일을 로드하고 드라이버를 register할 수 있도록 한다. registerbgidriver에 메모리상의 주소가 한번 전달되면 initgraph는 등록된 드라이버를 사용 한다. 사용자가 등록한 드라이버는 디스크로부터 heap에 읽혀지거나 .obj 파일로 변환되어 (BGIOBJ 파일을 이용) .EXE 파일에 링크될 수 있다.

registerbgidriver를 호출하면 그래픽 시스템은 드라이버가 링크시에 driver가 지정한 위치에 들어 있음을 알게 된다. 이 루틴은 지정된 드라이버에 대하여 링크-인 코드를 검사한다.

리 턴 값 만일 지정된 드라이버나 폰트에 문제가 있으면 registerbgidriver는 그래픽 에러 코드를 리턴하거나 드라이버 번호를 리턴한다. 만일 사용자가 제공한 드라이버를 등록시키면 registerbgidriver가 리턴하는 드라이버 번호를 필히 initgraph에 전달하여야 한다.

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `graphresult`,`initgraph`, `installuserdriver`,`registerbgifont`

쌇registerbgifont
#함수 REGISTERBGIFONT

기 능 링크-인 스트로크 폰트 코드를 등록한다.

구 문 #include <graphics.h>
int registerbgifont(void *font)(void);

프로토타입 `graphics.h`

기능 설명 registerbgifont를 호출함으로써 그래픽 시스템 상에서 font가 지정한 위치에 링크시 포함되는 폰트가 있음을 알려 준다.
이 루틴은 해당 폰트를 검색한 후 올바르면 내부 테이블에 등록한다.

자신의 폰트를 사용하고자 할 경우 registerbgifont의 결과를 폰트 번호가 사용될 settextstyle로 전달하여야 한다.

리 턴 값 만일 지정한 폰트가 유효하지 않은 경우 registerbgifont는 음수의 그래픽 에러 코드를 에러 코드를 리턴한다 그 외에는 등록된 폰트의 번호를 리턴하게 된다.

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `graphresult`,`initgraph`, `installuserdriver`,`registerbgifont`, `settextstyle`

쌇restorecrtmode
#함수 RESTORECRTMODE

기　　능　화면 모드를 initgraph에서 설정하기 전의 상태로 복귀한다.
구　　문　#include <graphics.h>
　　void far restorecrtmode(void);
프로토타입　`graphics.h`
기능 설명　initgraph에 의해 탐색된 원래의 비디오 모드로 복귀한다. 이 함수는 setgraphmode와 연계하여 텍스트 모드와 그래픽 모드를 전환하는데 사용될 수 있다.
　　textmode는 한 텍스트 모드에서 다른 텍스트 모드로 전환할 경우에 사용되기 때문에 이러한 용도로 써서는 않된다.
리 턴 값　없음
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참　　조　`getgraphmode`,`initgraph`,`setgraphmode`

쌇sector
#함수 SECTOR

기　　능　타원형의 파이 조각을 그린 후 내부를 칠한다
구　　문　#include <graphics.h>
　　void far sector(int x,int y
　　int stangle,int endangle,
　　int xradius,int yradius);
프로토타입　`graphics.h`
기능 설명　(x,y)를 중심으로 xradius를 수평 반지름으로 yradius를 수직 반지름으로 하여 stangle부터 endangle 까지의 타원형 파이 조각을 그린후 내부를 칠한다. 외곽선은 현재의 그리고 있는 색으로 내부는 현재의 fill 패턴과 색상으로 채워진다.
리 턴 값　없음
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참　　조

쌿setactivepage
#함수 SETACTIVEPAGE

기　　능　그래픽 출력을 행하는 페이지를 설정한다.
구　　문　#include <grahpcsi.h>
　　void far setactivepage(int page);
프로토타입　`graphics.h`
기능 설명　그래픽출력이 이루어질 그래픽 페이지를 page
　　로 설정한다. 그래픽 페이지는 화면에 보일수
　　도 화면에 보이지 않을 수 도 있는데 그 여부
　　는 사용자 시스템이 얼마나 많은 그래픽 페이
　　지가 있는지에 좌우된다.
리 턴 값　없음
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘
　　IBM-PC 와 호환기종에 한한다.
참　　조　`setvisualpage`

쌿setallpalette
#함수　SETALLPALETTE

기　　능　모든 팔레트 색상을 지정한 색으로 바꾼다.
구　　문　#include <graphics.h>
　　void far setallpalette(struct palettetype
　　far *palette);
프로토타입　`graphics.h`
기능 설명　palette가 지정한 palettetype 구조체에 있는
　　대로 현재의 팔레트를 설정한다. 사용자는 se
　　tallpalette를 사용하여 EGA/VGA 팔레트 색상
　　을 일부 혹은 전부 변경할 수 있다.
　　setallpalette에 사용되는 MAXCOLORS 상수와
　　palette 구조체는 다음과 같다.

```
#define MAXCOLORS 15
struct palettetype
{
  unsigned char size;
  signed char colors[MAXCOLORS + 1];
};
```

　　size에는 현재의 모드에서 그래픽 드라이버에
　　대한 해당 팔레트 내의 색상번호를 넣는다.

colors는 size바이트의 매열인데 해당 팔레트 내의 각 항목에 대한 실제 색상의 번호가 들어 있다. 만일 colors의 요소중 하나가 -1 이면 그에 해당하는 색상은 변하지 않는다. setallpalette는 IBM-8514에서는 쓸 수 없다.

리 턴 값 setallpalette에 부적당한 입력이 전달되면 graphresult는 -11을 리턴한다. 현재의 팔레트는 변경되지 않는다.

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `getpalette`,`graphresult`,`setbkcolor`, `setcolor`,`setpalette`

쌿setaspectratio
#함수 SETASPECTRATIO

기 능 디폴트 종횡비(aspect ratio)의 수정 인수를 변경한다.

구 문 #include <graphics.h>
void far setaspectraio(int xasp,int yasp);

프로토타입 `graphics.h`

기능 설명 해당 그래픽 시스템의 디폴트 종횡비를 바꾼다. 종횡비는 시스템이 정확한 원을 그리도록 하는데 사용된다.

만일 원이 타원으로 그려지면 이것은 모니터가 제대로 조정되지 않은 것인데 setaspectratio를 사용하여 소프트웨어적 으로 해결할 수 있다.

리 턴 값 없음

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `circle`,`getaspectratio`

쌿setbkcolor
#함수 SETBKCOLOR

기 능 해당 팔레트에 사용하는 배경색을 정한다.

구 문 #include <graphics.h>
void far setbkcolor(int color);

프로토타입 `graphics.h`

기능 설명 배경색을 color로 설정한다. 예를 들어 사용자가 배경을 파랗게 칠하기 원한다면 다음과 같이 호출한다.

setbkcolor(BLUE);

리 턴 값 없음

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `getbkcolor`,`setallpalette`, `setcolor`,`setpalette`

쌇setcolor

#함수 SETCOLOR

기 능 해당 팔레트를 사용하여 현재 지정된 색상을 설정한다.

구 문 #include <graphics.h>
void far setcolor(int color);

프로토타입 `graphics.h`

기능 설명 그리고자 하는 색상을 color로 설정한다. color의 범위는 0에서 getmaxcolor까지이다.

리 턴 값 없음

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `getcolor`,`getmaxcolor`,`setallpalette`, `setbkcolor`,`setpalette`

쌇setfillpattern

#함수 SETFILLPATTERN

기 능 사용자가 정의한 필(fill) 형태를 선택한다.

구 문 #include <graphics.h>
void far setfillpattern(char far *upattern ,int color);

프로토타입 `graphics.h`

기능 설명 setfillstyle 과 유사하지만 사전 지정 형태 대신에 사용자가 지정하는 8x8 픽셀의 형태를 설정할 수 있는 점에서 다르다.
upattern은 일련의 8바이트에 대한 포인터이다. 각각의 바이트는 8 픽셀을 의미한다.

리 턴 값 없음

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘
IBM-PC 와 호환기종에 한한다.
참 조 `getfillpattern`,`getfillsettings`,
`setcolor`,`setfillstyle`

쌇setfillstyle
#함수 SETFILLSTYLE

기 능 fill 형태와 색상을 설정한다.
구 문 #include <graphics.h>
void far setfillstyle(int pattern,
int color);
프로토타입 `graphics.h`
기능 설명 fill형태와 색상을 설정한다. 사용자 정의 필
형태를 사용하기 위해서는 setfillstyle에 pa
ttern을 대신 setfillpattern을 호출한다.
EMPTY_FILL을 제외한 나머지는 현재 지정된
필 색상으로 채운다. EMPTY_FILL은 배경색으
로 채운다.
setfillstyle에 부당한 입력이 전달되면 grap
hresult는 -11을 리턴하고, 채욱 색상은 변하
지 않는다.
리 턴 값 없음
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘
IBM-PC 와 호환기종에 한한다.
참 조 `bar`,`bar3d`,`fillpoly`,`floodfill`,
`getfillsettings`,`graphresult`,`pieslice`,
`sector`,`setfillpattern`

쌇setgraphbufsize
#함수 SETGRAPHBUFSIZE

기 능 그래픽 내부 버퍼 크기를 변경한다.
구 문 #include <graphics.h>
unsigned far setgraphbusize(unsigned
bufsize);
프로토타입 `graphics.h`
기능 설명 그래픽 루틴의 일부(floodfill 등)는 initgra
ph가 호출될 때 할당하고 closegraph가 호출
되면 해제되는 메모리 버퍼를 사용한다. 이
버퍼는 _graphgetmem에 의해서 할당되는데 디

폴트 크기는 4096바이트 이다.

사용자는 이 버퍼의 크기를 줄이거나 늘이기 위해 setgraphbufsize를 호출할 수 있다. setgraphbufsize는 항상 initgraph보다 먼저 호출되어야 하며 그렇지 않으면 다음 closegraph 호출시 까지 setgraphbufsize호출은 모두 무시된다.

리 턴 값 내부 버퍼의 변경된 크기를 리턴한다.

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `closegraph`,`_graphfreemem`, `_graphgetmem`,`initgraph`,`sector`

쌇setgraphmode

#함수 SETGRAPHMODE

기 능 시스템을 정해진 그래픽 모드로 변환하고 화면을 초기화 한다.

구 문 #include <graphics.h>
void far setgraphmode(int mode);

프로토타입 `graphics.h`

기능 설명 initgraph에 설정된 디폴트 그래픽 모드와 다른 그래픽 모드를 선택한다. 이 때 mode 값은 지정된 그래픽 드라이버에 합당한 값이어야 한다.
setgraphmode는 화면을 초기 상태로 하고 모든 변수의 값을 초기상태로 바꾼다.(CP,팔레트,색상,뷰포트 등)

리 턴 값 없음

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `setgraphmode`,`setmoderange`, `graphresult`,`initgraph`,`restorecrtmode`

쌇setlinestyle

#함수 SETLINESTYLE

기 능 선의 굵기와 형태를 정한다.

구 문 #include <graphics.h>
void far setlinestyle(int linestyle,

unsigned upattern,int thickness);

프로토타입 `graphics.h`

기능 설명 line,lineto,rectangle,drawpoly 등의 선의 굵기와 형태를 정한다.

linestyle 은 선이 그려지는 스타일을 실선, 점선, 일점 쇄선, 파선 등으로 지정한다. graphics.h 에 정의되어 있는 line_style 은 다음과 같다.

이름	값	내용
SOLID_LINE	0	실선
DOTTED_LINE	1	점선
CENTER_LINE	2	일점 쇄선
DASHED_LINE	3	파선
USERBIT_LINE	4	사용자 선

thickness 는 선의 굵기를 지정하며 다음과 같이 지정된다.

이름	값	내용
NORM_WIDTH	1	1 픽셀
RHICK_WIDTH	3	3 픽셀

리 턴 값 없음

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `bar3d`,`getlinesettings`,`graphresult`, `line`,`linerel`,`lineto`,`rectangle`

쌇setpalette

#함수 SETPALETTE

기 능 사용자가 팔레트 색상을 지정한다.

구 문 #include <graphics.h>
void far setpalette(int colornum,
int color);

프로토타입 `graphics.h`

기능 설명 팔레트 colornum 항목을 color 로 바꾼다. 예를 들어 setpalette(0,5)는 현재 팔레트의 첫 번째 색상을 색상 5로 바꾼다. 만일 size가 현재 팔레트가 가지는 최대 항목이면 colornum 값은 0에서 size-1까지 이다.

팔레트에 대한 변경은 화면에 즉시 결과가 나타난다. 하나의 팔레트가 바뀌어 지면 화면상의 그 색상은 모두 바뀌어진 색상으로 변하게 된다.

리 턴 값 setpalette의 전달된 입력이 부적당 하면 graphresult는 -11을 리턴한다.

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `getpalette`,`graphresult`,`setallpalette`, `setbkcolor`,`setcolor`,`setrgbpalette`

쌇setrgbpalette

#함수 SETRGBPALETTE

기 능 사용자가 색상을 지정할 수 있게한다.

구 문 #include <graphics.h>
void far setrgbpalette(int colornum,
int red,int green,int blue);

프로토타입 `graphics.h`

기능 설명 colornum은 팔레트에 로드될 항목을 설정하고 red, green, blue는 각각의 색조합을 나타낸다.

리 턴 값 없음

이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.

참 조 `setpalette`

쌇settextjustify

#함수 SETTEXTJUSTIFY

기 능 그래픽 텍스트 정렬을 설정한다.

구 문 #include <graphics.h>
void far settextjustify(int horiz,
int vert);

프로토타입 `graphics.h`

기능 설명 settextjustify를 호출한 뒤에 텍스트 출력은
CP 주위에 수평.수직으로 움직이며 정렬된다.
처음에 설정된 값은 LEFT_TEXT,TOP_TEXT이다.

settextjustify는 outtext에 의해 쓰여지는
문자에만 영향을 미치며 텍스트 모드와 스트
림 함수를 함께 사용할 수 없다.

리 턴 값 없음
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘
IBM-PC 와 호환기종에 한한다.
참 조 `gettextsettings`,`graphresult`,`outtext`,
`settextstyle`

쌿settextstyle
#함수 SETTEXTSTYLE

기 능 그래픽 출력을 위한 텍스트의 특징을 설정한
다.
구 문 #include <graphics.h>
void far settextstyle(int font,
int direction,int charsize);
프로토타입 `graphics.h`
기능 설명 텍스트의 폰트, 표시방향과 문자의 크기를 지
정한다. 이 함수의 호출은 outtext와 outtext
xy에 의해 출력되는 텍스트에 영향을 미친다.
settextstyle에 전달되는 font,direction,cha
rsize 파라미터는 다음과 같은 기능을 가지고
있다.

font : 하나의 8x8 폰트와 약간의 stroked 폰
트를 사용할 수 있다. 디폴트 폰트는
시스템에 내장되어 있으며 stroked 폰트는 디
스크의 각 .CHR 파일에 저장되어 있다.

direction : 지원되는 폰트의 방향은 수평(좌
에서 우로) 및 수직(반 시계 방
향으로 90도 회전) 텍스트 이다.
디폴트 방향은 HORIZ_DIR 이다.

이름 값 설명

HORIZ_DIR	0	좌에서 우로
VERT_DIR	1	하에서 상으로

charsize : 각 문자의 크기는 charsize를 이용 하여 확대할 수 있다. 만일 charsize가 0이 아니면 비트맵이나 stroked 폰트에 적용되지만 0이면 stroked 폰트에만 적용된다.

리 턴 값 없음
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참 조 `gettextsettings`,`graphresult`, `installuserfont`,`settextjustify`, `setusercharsize`,`texthight`,`textwidth`

쌿setusercharsize
#함수 SETUSERCHARSIZE

기 능 stroked 폰트의 문자폭과 높이를 사용자가 변경할 수 있도록 한다.
구 문 #include <graphics.h>
void far setusercharsize(int mulx,int divx, int muly,int divy);
프로토타입 `graphics.h`
기능 설명 setusercharsize를 이용하여 stroked 폰트의 크기를 조절할 수 있다. setusercharsize는 settextstyle에 의해 지정된다.

지정하는 폰트의 크기는 폭의 경우 mulx:divx 높이는 nuly,divy의 비례로 지정된다. 예를 들어 디폴트 폭의 2배, 높이를 15배로 하려면 각각의 값은 다음과 같다.

mulx = 2; divx = 1;
muly = 3; divy = 2;

리 턴 값 없음
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참 조 `gettextsettings`,`graphresult`,

`settextstyle`

쌇setviewport
#함수 SETVIEWPORT

기　　능　그래픽 출력을 위한 뷰포트를 설정한다.
구　　문　#include <graphics.h>
　　void far setviewport(int left,int top,
　　int right,int bottom,int clip);
프로토타입　`graphics.h`
기능 설명　그래픽 출력을 위한 새로운 뷰포트를 설정한다.
　　뷰포트는 좌상귀(left,top)와 우하귀(right,bottom)을 장방향으로 하는 화면상의 절대좌표를 나타낸다.
리 턴 값　setviewport에 의해 설정된 값이 적절하지 않으면 graphresult는 -11을 리턴한다. 현재의 뷰포트 설정은 변하지 않는다.
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참　　조　`clearviewport`,`setviewsettings`,
　　`graphresult`

쌇setvisualpage
#함수 SETVISUALPAGE

기　　능　화면에 나타날 그래픽 페이지를 설정한다.
구　　문　#include <graphics.h>
　　void far setvisualpage(int page);
프로토타입　`graphics.h`
기능 설명　page를 화면에 나타날 그래픽 페이지로 선택한다.
리 턴 값　없음
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참　　조　`graphresult`,`setactivepage`

쌇setwritemode
#함수 SETWRITEMODE

기　　능　그래픽 모드에서 선을 긋는 모드를 설정한다.

구　　문　#include <graphics.h>
　　void far setwritemode(int mode);
프로토타입　`graphics.h`
기능 설명　다음과 같은 상수가 정의되어 있다.
　　COPY_PUT = 0
　　XOR_PUT = 1
　　각 상수는 선을 그을때 화면상에 대응되는 이
　　진 연산의 방법을 지정한다. COPY_PUT은 어셈
　　블리의 MOV를 사용하여 화면상의 어떤것이든
　　겹쳐서 선을 그린다.

　　XOR_PUT은 XOR 명령을 사용하므로 이전에 존
　　재 하였던 선은 지워지고 선이 없었던 부분만
　　그려진다.
리 턴 값　없음
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘
　　IBM-PC 와 호환기종에 한한다.
참　　조　`drawpoly`,`line`,`linerel`,`lineto`,
　　`putimage`

쌇texthight
#함수 TEXTHIGHT

기　　능　문자열의 높이를 픽셀단위로 리턴한다.
구　　문　#include <graphics.h>
　　int far texthight(char far *textstring);
프로토타입　`graphics.h`
기능 설명　그래픽 함수 texthight는 현재 폰트의 크기와
　　확대 비율을 가지고 문자열 textstring의 높
　　이를 계산하여 픽셀단위로 리턴한다.
리 턴 값　문자열의 높이를 픽셀단위로 리턴한다.
이 식 성　지원되는 그래픽 디스플레이 어댑터를 갖춘
　　IBM-PC 와 호환기종에 한한다.
참　　조　`gettextsettings`,`outtext`,`outtextxy`,
　　`settextstyle`,`textwidth`

쌇textwidth
#함수 TEXTWIDTH

기　　능　문자열의 넓이를 픽셀단위로 리턴한다.
구　　문　#include <graphics.h>

int far textwidth(char far *textstring);
프로토타입 `graphics.h`
기능 설명 그래픽 함수 texthight는 현재 폰트의 크기와 확대 비율을 가지고 문자열 textstring의 넓이를 계산하여 픽셀단위로 리턴한다.
리 턴 값 문자열의 넓이를 픽셀단위로 리턴한다.
이 식 성 지원되는 그래픽 디스플레이 어댑터를 갖춘 IBM-PC 와 호환기종에 한한다.
참 조 `gettextsettings`,`outtext`,`outtextxy`, `settextstyle`,`texthight`

쌇io.h
#헤더파일 IO.H

함수

`access` `_chmod` `chmod`
`chsize` `_close` `close`
`_creat` `creat` `creatnew`
`creattemp` `dup` `dup2`
`eof` `filelength` `getftime`
`ioctl` `isatty` `lock`
`lseek` `_open` `open`
`_read` `read` `setftime`
`setmode` `sopen` `tell`
`unlink` `unlock` `_write`
`write`

쌇access
#함수 ACCESS

기 능 파일의 엑세스 기능을 결정한다.
구 문 int access(const char *filename, int amode);
프로토타입 `io.h`
기능 설명 파일이 존재하는지, 그리고 읽고 쓸 수 있는지, 실행 시킬 수 있는지의 여부를 결정하기 위해서 filename에 의해서 명명된 파일을 검사한다.

amode 값의 목록은 다음과 같다.
---

06 일고 쓸수 있는지를 검사한다.
04 읽을 수 있는지를 검사한다.
02 쓸 수 있는지를 검사한다.
01 실행할 수 있는지를 검사한다.
00 파일의 존재를 검사한다

리 턴 값 요청된 억세스가 허용된 경우에는 0을 리턴한다. 그렇지 않으면 -1을 리턴하고`errno`은 다음과 같이 설정된다.

ENOENT File or pathname not found
EACCESS Permission denied

이 식 성 UNIX 시스템에서 사용할 수 있다.
참 조 `chmod`,`fstat`,`stat`

쌿_chmod
#함수 _CHMOD

기 능 파일 억세스 모드를 바꾼다.
구 문 #include <dos.h>
#include <io.h>
int _chmod(const char *path,
int func[,int attrib]);
프로토타입 `io.h`
기능 설명 DOS의 파일속성을 불러오거나 설정한다. func가 0이면 함수는 현재 속성을 리턴하고 1인 경우는 그 속성이 attrib에 설정된다. attrib 눈 다음과 같은 기호 상수중(dos.h 에 정의) 하나가 된다.

FA_RDONLY 읽어 들이기만 한다.
FA_HIDDEN 숨겨진 파일이 된다.
FA_SYSTEM 시스템 파일이 된다.

리 턴 값 에러 없이 수행되면 파일의 속성값을 리턴하고 에러가 발생하면 -1이 리턴된다. 이 때 `errno`는 다음과 같이 설정된다.

ENOENT Path or filename not found
EACCES Permission denied

이 식 성 DOS에 한한다.
참 조 `chmode`,`_creat`

쌇chmod
#함수 CHMOD

기　　능　파일의 억세스 모드를 변경한다.
구　　문　#include <sys\stat.h>
　　int chmode(const char *path,int amode);
프로토타입　`io.h`
기능 설명　amode 에서 주어진 mask에 따라서 filename의 속성을 변경한다. amode는 sys\stat.h에서 정의된 S_IWRITE와 S_IREAD의 기호 상수 중 하나 내지 둘을 포함할 수 있다.

amode의 값	억세스 허용
S_IWRITE	쓰기 가능
S_IREAD	읽기 가능
S_IREAD|S_IWRITE	쓰기/읽기 가능

리 턴 값　에러없이 수행되면 0을 리턴하고 에러가 발생하면 -1을 리턴한다. 이 때 errno은 다음과 같이 설정된다.

　　ENOENT　Path or filename not found
　　EACCES　Permission denied
이 식 성　UNIX 시스템에서 사용할 수 있다.
참　　조　`access`,`_chmod`,`fstat`,`open`,`sopen`, `stat`

쌇chsize
#함수 CHSIZE

기　　능　파일의 크기를 변경한다.
구　　문　int chsize(int handle,long size);
프로토타입　`io.h`
기능 설명　handle과 연관된 파일 크기를 변경한다. 파일의 원래 크기와 비교되는 size 값에 따라 파일이 잘리거나 확장된다. 파일의 모드는 쓰기가 가능해야 한다. chsize가 파일을 확장하는 경우에는 null 문자(0)을 붙여야 하고 파일을 자

르는 경우 end-of-file 이후의 데이타는 분실
된다.
리 턴 값 에러없이 수행되면 0을 리턴하고 에러가 발생
하면 -1을 리턴한다. 이 때 errno은 다음과
같이 설정된다.

EACCES Permission denied
EBADF Bad file number
ENOSPC UNIX - not DOS
이 식 성 DOS에 한한다.
참 조 `close`,`_creat`,`creat`,`open`

쌿_close
#함수 _CLOSE

기 능 파일을 닫는다.
구 문 int _close(int handle);
프로토타입 `io.h`
기능 설명 handle과 연관된 파일을 닫는다. handle은 _c
reat,creat,creatnew,creattemp,dup,dup2,_op
en,open 호출로 부터 얻어진 파일 핸들이다.
리 턴 값 에러없이 수행되면 0을 리턴하고 에러가 발생
하면 -1을 리턴한다. 이 때 errno은 다음과
같이 설정된다.

EBADF Bad file number
이 식 성 DOS에 한한다.
참 조 `close`,`creat`,`open`,`read`,`write`

쌿close
#함수 CLOSE

기 능 파일을 닫는다.
구 문 int close(int handle);
프로토타입 `io.h`
기능 설명 handle과 연관된 파일을 닫는다. handle은 _c
reat,creat,creatnew,creattemp,dup,dup2,_op
en,open 호출로 부터 얻어진 파일 핸들이다.
리 턴 값 에러없이 수행되면 0을 리턴하고 에러가 발생
하면 -1을 리턴한다. 이 때 errno은 다음과
같이 설정된다.

EBADF Bad file number
이 식 성 UNIX 시스템상에서 사용할 수 있다.
참 조 `_close`,`creat`,`open`,`read`,`write`

쌇creat
#함수 CREAT

기 능 새로운 파일을 작성하거나 기존의 파일을 다시 작성한다.
구 문 #include <sys\stat.h>
int creat(const char *path, int amode);
프로토타입 `io.h`
기능 설명 새로운 파일을 작성하거나 path로 주어진 기존의 파일을 재작성한다. amode는 새로이 작성되는 파일에만 적용된다.

crear는 파일이 존재하고 쓰기 모드가 설정되어 있는경우 파일모드는 그대로 두고 파일의 길이를 0으로 만든다. 그러나 같은 파일이 존재 하면서 읽기 어트리뷰트가 설정되어 있는 경우 creat 호출을 에러를 발생한다.

creat 호출은 억세스 모드 워드인 amode의 S_IWRITE 비트만을 검사한다. 이 비트가 1이면 파일은 쓰기 가능한 것이 되고 0이면 파일은 읽기만 허용된다. 그리고 다른 MS-DOS 어트리뷰트는 0으로 설정된다.

amode의 값	억세스 허용
S_IWRITE	쓰기 가능
S_IREAD	읽기 가능
S_IREAD|S_IWRITE	쓰기/읽기 가능

리 턴 값 에러없이 수행되면 0 이상의 새로운 파일의 핸들을 리턴한다. 에러가 발생하면 -1을 리턴한다, 이 때`errno`은 _creat와 같이 설정된다.
이 식 성 UNIX 시스템에서 사용할 수 있다.
참 조 `chmod`,`chsize`,`close`,`_creat`,

`creatnew`,`creattemp`,`dup`,`dup2`,
`fopen`,`open`,`sopen`,`write`

쌇_creat
#함수 _CREAT

기 능 새로운 파일을 작성하거나 기존의 파일을 다시 작성한다.
구 문 include <dos.h>
int _creat(const char *path, int attrib);
프로토타입 `io.h`
기능 설명 DOS의 attribute 워드를 받는다. 어떠한 어트리뷰트 비트도 이 함수에서 설정될 수 있다. 파일은 언제나 이진 모드로 열린다. 정상적으로 파일이 작성되면 파일 포인터가 파일의 선두에 설정되고 읽기와 쓰기 가능한 파일로 열린다. 파일이 이미 존재하는 경우에는 파일의 크기는 다시 0으로 설정된다.

FA_RDONLY 읽어 들이기만 한다.
FA_HIDDEN 숨겨진 파일이 된다.
FA_SYSTEM 시스템 파일이 된다.

리 턴 값 에러 없이 수행되면 파일의 핸들값을 리턴하고 에러가 발생하면 -1이 리턴된다. 이 때 `errno`는 다음과 같이 설정된다.

ENOENT Path or filename not found
EMFILE Too many open file
EACCES Permission denied

이 식 성 DOS에 한한다.
참 조 `_chmod`,`chsize`,`_close`,`close`,`creat`,
`creatnew`,`creattemp`

쌇creatnew
#함수 CREATNEW

기 능 새로운 파일을 작성한다.
구 문 #include <dos.h>
int creatnew(const char *path,int attrib);
프로토타입 `io.h`

기능 설명 _creat 와 동일하나 예외적으로 파일이 존재하는 경우 파일은 그대로 두고 에러를 리턴한다. creatnew의 인수는 다음중의 하나이다.

FA_RDONLY 읽어 들이기만 한다.
FA_HIDDEN 숨겨진 파일이 된다.
FA_SYSTEM 시스템 파일이 된다.

리 턴 값 에러 없이 수행되면 파일의 핸들값을 리턴하고 에러가 발생하면 -1이 리턴된다. 이 때 `errno`는 다음과 같이 설정된다.

EEXIST File aleady exists
ENOENT Path or filename not found
EMFILE Too many open file
EACCES Permission denied

이 식 성 DOS에 한한다.
참 조 `close`,`_creat`,`creat`,`creattemp`,`dup`, `open`

쌿creattemp
#함수 CREATTEMP

기 능 path가 걸린 디렉토리에 새로운 파일을 작성한다.
구 문 #include <dos.h>
int creattemp(char *path, int attrib);
프로토타입 `io.h`
기능 설명 패스명은 역슬래시(\)로 끝난다. path가 걸린 디렉토리에 유일한 파일명이 선택되며, 새롭게 작성된 파일명은 지정된 path 문자열에 저장 된다. 이 때 패스는 새롭게 작성된 파일을 수용할 수 있을 정도로 충분한 공간을 가지고 있어야 한다. 파일은 프로그램이 종료되어도 자동으로 삭제되지 않는다.
리 턴 값 에러 없이 수행되면 파일의 핸들값을 리턴하고 에러가 발생하면 -1이 리턴된다. 이 때 `errno`는 다음과 같이 설정된다.

ENOENT Path or filename not found
EMFILE Too many open file

EACCES Permission denied

이 식 성 DOS에 한한다.
참 조 `close`,`_creat`,`creat`,`creatnew`,`dup`,
`open`

쌇dup
#함수 DUP

기 능 파일 handle을 복사한다.
구 문 int dup(int handle);
프로토타입 `io.h`
기능 설명 일반적으로 다음과 같은 원래 파일 handle을
갖는 새로운 파일 handle을 리턴한다.

* 동일한 파일이나 디바이스
* 동일한 파일 포인터
* 동일한 억세스 모드

리 턴 값 에러가 없으면 dup는 양의 정수인 새로운 파
일 handle을 리턴하고 에러가 발생하면 -1을
리턴한다. 에러인 경우`errno`은 다음과 같이
설정된다.

EMFILE Too many open file
EBADF Bad file number
이 식 성 모든 UNIX 시스템에서 사용될 수 있다.
참 조 `_close`,`close`,`_creat`,`creat`,
`creatnew`,`creattemp`,`dup2`,`fopen`,
`_open`,`open`

쌇dup2
#함수 DUP2

기 능 파일 handle을 복사한다.
구 문 int dup(int handle);
프로토타입 `io.h`
기능 설명 일반적으로 다음과 같은 원래 파일 handle을
갖는 새로운 파일 handle을 리턴한다.

* 동일한 파일이나 디바이스

* 동일한 파일 포인터
* 동일한 억세스 모드

리 턴 값 에러가 없으면 dup는 양의 정수인 새로운 파일 handle을 리턴하고 에러가 발생하면 -1을 리턴한다. 에러인 경우`errno`은 다음과 같이 설정된다.

EMFILE Too many open file
EBADF Bad file number

이 식 성 몇몇 UNIX 시스템에서는 사용될 수 있지만 SYSTEM III 에서는 사용할 수 없다.

참 조 `_close`,`close`,`_creat`,`creat`, `creatnew`,`creattemp`,`dup`,`fopen`, `_open`,`open`

쌿eof
#함수 EOF

기 능 파일의 종료(end-of-file) 여부를 검사한다.

구 문 int eof(int handle);

프로토타입 `io.h`

기능 설명 파일 handle과 연관된 파일에 EOF 에 와 있는지 검사한다.

리 턴 값 현재의 위치가 파일의 끝이면 eof는 1을 리턴하고 그렇지 않으면 0을 리턴한다. 1을 리턴하면 에러가 발생한 것이며 이때`errno`은 다음과 같이 설정된다.

EBADF Bad file number

참 조 `clearerr`,`feof`,`ferror`,`perror`

쌿filelength
#함수 FILELENGTH

기 능 파일의 크기를 바이트 단위로 갖는다.

구 문 #include <io.h>
long filelength(int handle);

프로토타입 `io.h`

기능 설명 handle과 연관된 파일의 길이를 바이트 단위로 리턴한다.

리 턴 값 에러없이 수행되면 long 형 값인 파일의 크기를 바이트로 리턴한다. 에러 발생시 에는 -1

을 리턴한다.
참 조 `fopen`,`lseek`,`open`

쌇getftime
#함수 GETFTIME

기 능 파일의 시간과 일자를 갖는다.
구 문 #include <io.h>
int getftime(int hangle,
struct ftime *ftimep);
프로토타입 `io.h`
기능 설명 오픈되어 있는 handle과 연관된 디스크 파일에 대한 파일 시간과 일자를 점검한다. ftime 구조체는 다음과 같이 정의된다.

```
struct ftime {
    unsigned ft_tsec  : 5;
    unsigned ft_min   : 6;
    unsigned ft_hour  : 5;
    unsigned ft_day   : 5;
    unsigned ft_month : 4;
    unsigned ft_year  : 7;
};
```

리 턴 값 에러가 없으면 0이 리턴되며, 에러가 발생하면 -1을 리턴한다.
이 식 성 DOS에 한한다.
참 조 `open`,`setftime`

쌇ioctl
#함수 IOCTL

기 능 IO 디바이스를 제어한다.
구 문 int ioctl(int handle,int func,
[,void *argdx,int argcx]);
프로토타입 `io.h`
기능 설명 DOS 호출 0x44(IOCTL)에 직접 인터페이스 한다. 다음과 같은 func값에 따라서 정확한 함수 기능이 실행된다.

0 디바이스 정보를 가져 온다.
1 디바이스 정보를 argdx에 설정한다.

2 argdx가 가리키는 어드레스에 argcx 바이트를 읽어온다.
3 argdx가 가리키는 어드레스로부터 argcx 바이트로 작성한다.
4 2와 같다. 다만 handle을 드라이브 번호로 취급한다. (0=디폴트, 1=A 드라이브)
5 3와 같다. 다만 handle을 드라이브 번호로 취급한다. (0=디폴트, 1=A 드라이브)
6 입력 상태를 가져 온다.
7 출력 상태를 가져 온다.
8 메시지 교환 가능성 여부를 검사한다.
9 재시도 횟수를 설정한다.

이 식 성 UNIX 시스템상에서 사용할 수 있지만 파라미터나 함수 기능으로는 사용할 수 없다.

쌇isatty
#함수 ISATTY

기 능 디바이스 형태를 검사한다.
구 문 int isatty(int handle);
프로토타입 `io.h`
기능 설명 handle이 가리키는 문자 디바이스의 형태를 검사한다.

* ternimal 터미널
* console 콘솔
* printer 프린터
* serial port 시리얼 포트

리 턴 값 디바이스가 문자 디바이스인 경우 isatty 는 0 이외의 정수를 리턴한다. 이와 같은 디바이스가 아닌 경우 0을 리턴한다.

쌇lock
#함수 LOCK

기 능 파일의 공유(sharing) 잠김을 설정한다.
구 문 int lock(int handle,long offset, long length);
프로토타입 `io.h`
기능 설명 파일 공유 메카니즘과의 인터페이스를 제공한다. lock은 파일이 오버랩되지 않은 임의의

영역에 위치할 수 있다. 잠겨진 영역에 읽기
나 쓰기를 시도하는 프로그램은 세 번까지 재
시도를 하게 된다. 이 세번의 재시도가 실패
하면 이 호출은 에러가 된다.

리 턴 값 에러가 없으면 0을 리턴한다. 에러시에는 -1
을 리턴한다.

이 식 성 DOS에 한한다.

참 조 `open`,`sopen`,`unlock`

쌿lseek

#함수 LSEEK

기 능 파일 포인터의 이동

구 문 #include <io.h>
long lseek(int handle,long offset,
int fromwhere);

프로토타입 `io.h`

기능 설명 handle과 연관된 파일 포인터를 fromwhere로
주어진 파일 위치로 부터 offset 바이트 이동
한 새로운 위치로 설정한다. fromwhere는 다
음 세 상수중 하나를 가진다.

fromwhere	파일 위치
SEEK_SET(0)	파일의 시작 부분
SEEK_CUR(1)	현재 파일 포인터의 위치
SEEK_END(2)	파일의 끝

리 턴 값 파일의 시작 부분에서 바이트로 측정된 새로
운 파일 포인터의 옵셋을 리턴한다. 에러 발
생시 lseek은 -1L을 리턴하고`errno`은 다음
중 하나로 설정된다.

EBADF Bad file number
EINVAL Invalid argument

이 식 성 모든 UNIX 시스템상에서 사용할 수 있다.

참 조 `filelength`,`fseek`,`ftell`,`sopen`,
`_write`,`write`

쌿_open

#함수 _OPEN

기　　능　쓰기 또는 읽기용 으로 파일을 개방한다.
구　　문　#include <fcntl.h>
　　int _open(const char *filename,
　　int oflags);
프로토타입　`io.h`
기능 설명　filename에 의해 지정된 파일을 개방하고 ofl
　　ag의 값에 의해 결정된 쓰기/읽기용으로 파일
　　을 준비한다.
　　_open에 대해서 DOS 2.x 에서 oflag의 값은 O
　　_RDONLY,O_WRONLY, 그리고 ORDWR로 제한된다.
리 턴 값　에러가 없으면 _open은 음이 아닌 정수(파일
　　handle)를 리턴한다. 에러시에는 -1을 리턴하
　　고`errno`은 다음 중 하나로 설정된다.

　　ENOENT　Path or filename not found
　　EMFILE　Too many open file
　　EACCES　Permission denied
　　EINVAC　Invalid access code

이 식 성　DOS에 한한다.
참　　조　`open`,`_read`,`sopen`

쌇open
#함수 OPEN

기　　능　쓰기와 읽기용으로 파일을 개방한다.
구　　문　#include <fcntl.h>
　　#include <sys\stat.h>
　　int open(const char *path, int access,
　　[, unsigned mode]);
프로토타입　`io.h`
기능 설명　path에 지정된 파일을 개방하고 access값에
　　의해 쓰기와 읽기용으로 준비한다. 특정 모드
　　로 파일을 개방하기 위해서 사용자는 fmode로
　　지정하거나 원하는 변환 모드와 함께 OR로 연
　　결된 O_CREAT와 O_TRUNC 옵션으로 open을 호
　　출한다.

　　O_NDELAY　UNIX 호환용, 여기서는 사용되지

않는다.
O_RDONLY 읽기 전용으로 개방
O_WRONLY 쓰기 전용으로 개방
O_RDWR 읽기와 쓰기용으로 개방
O_APPEND 이 플래그가 설정된 경우 파일 포
인터가 파일끝 으로 설정된다.
O_CREAT 이 플래그는 파일이 존재할 경우
영향이 없다. 파일이 존재 하지 않
을 경우 파일이 만들어 지고 mode
비트가 파일의 속성을 결정하는데
사용된다
O_TRUNC 파일이 존재하는 경우 파일의 크기
는 0으로 되지만, 파일의 속성은
변경되지 않는다.
O_EXCL O_CREAT 와 같이 사용된다. 파일이
존재하는 경우 에러가 리턴된다.
O_BINARY 파일을 이진 모드로 개방한다.
O_TEXT 파일을 텍스트 모드로 개방한다.

리 턴 값 에러가 없으면 open은 음이아닌 정수(handle)
을 리턴한다. 에러시에는 -1을 리턴하며
`errno`은 다음중 하나로 설정된다.

ENOENT Path or filename not found
EMFILE Too many open file
EACCES Permission denied
EINVAC Invalid access code

이 식 성 UNIX 시스템상에서 사용할 수 있다.
참 조 `chmod`,`chsize`,`close`,`creat`,
`creatnew`,`creattemp`,`dup`,`dup2`,
`fdopen`,`filelength`,`fopen`,`freopen`,
`getftime`,`lock`,`_open`,`read`,`sopen`,
`_write`

쌿_read
#함수 _READ

기 능 파일을 읽는다.
구 문 int _read(int handle, void *buf,
unsigned len);

프로토타입 `io.h`
기능 설명 handle과 연관된 file에서 len 바이트만큼 bu
f가 지정한 버퍼에 읽어들인다. _read는 바로
DOS의 읽기 명령을 호출한다. _read는 텍스트
모드로 개방된 파일을 읽을때 캐리지 리턴을
제거하지 않는다.

_read가 읽을 수 있는 최대 길이는 65535바이
트 이다. 65536(0xffff)는 -1과 같기 때문에
-1은 함수의 에러를 의미한다.
리 턴 값 에러가 없으면 _read는 읽어들인 바이트 수를
리턴한다. 파일의 종료에서는 0을, 에러발생
시는 -1을 리턴한다. 이때`errno`는 다음과
같이 설정된다.

EBADF Bad file number
EACCES Permission denied

이 식 성 DOS에 한한다.
참 조 `_open`,`read`,`_write`

쌿read
#함수 READ

기 능 파일을 읽는다.
구 문 int read(int handle, void *buf,
unsigned len);
프로토타입 `io.h`
기능 설명 handle과 연관된 file에서 len바이트만큼 buf
가 가리키는 버퍼에 읽어들인다.
파일이 텍스트 모드로 열린 경우는 캐리지 리
턴을 제거하고 파일의 끝에 도달하면 EOF 를
보고한다.

read가 읽을 수 있는 최대 길이는 65535바이
트 이다. 65536(0xffff)는 -1과 같기 때문에
-1은 함수의 에러를 의미한다.

리 턴 값 에러가 없으면 _read는 읽어들인 바이트 수를
리턴한다. 파일의 종료에서는 0을, 에러발생
시는 -1을 리턴한다. 이때`errno`는 다음과

같이 설정된다.

EBADF Bad file number
EACCES Permission denied

이 식 성 UNIX 시스템상에서 사용할 수 있다.
참 조 `open`,`_read`,`write`

쌿setftime
#함수 SETFTIME

기 능 파일의 날짜와 시간을 설정한다.
구 문 #include <io.h>
int setftime(int handle, struct ftime
*ftimep);
프로토타입 `io.h`
기능 설명 열려진 파일의 핸들을 handle로 전달받아서, 그 파일의 날짜와 시간을 ftimep가 가리키는 ftime 구조체에 설정된 내용대로 고친다.

ftime 구조는 다음과 같이 정의된다.

```
struct ftime {
 unsigned ft_tsec        초
 unsigned ft_min         분
 unsigned ft_hour        시
 unsigned ft_day         일
 unsigned ft_month       월
 unsigned ft_year        년 - 1980
};
```

리 턴 값 에러가 없으면 0을 리턴한다. 에러 발생시 -1을 리턴하고`errno`는 아래값 중 하나로 설정된다.

EINVFNC Invalid function number
EBADF Bad file number

이 식 성 DOS에 한한다.
참 조 `getftime`

쌿setmode
#함수 SETMODE

기 능 열린 파일의 모드를 설정한다.

구　　문　#include <fcntl.h>
　　int setmode(int handle,int amode);
프로토타입　`io.h`
기능 설명　handle과 연관된 파일의 모드를 이진이나 텍
　　스트 모드로 설정한다. 인수 amode는 O_BINA
　　RY 또는 O_TEXT 둘중의 하나만을 가져야 한
　　다.
리 턴 값　에러가 없으면 0을 리턴한다. 에러 발생시 -1
　　을 리턴하고`errno`은 아래와 같이 설정된다.

　　EINVAL　인수가 부적당하다.
이 식 성　UNIX 시스템상에서 사용할 수 있다.
참　　조　`_creat`,`creat`,`_open`,`open`

쌿sopen
#함수 SOPEN

기　　능　공유 파일을 연다.
구　　문　#include <fcntl.h>
　　#include <sys\stat.h>
　　#include <share.h>
　　#include <io.h>
　　int sopen (char *path,int access,
　　int shflag,int mode);
프로토타입　`io.h`
기능 설명　path 로 지정된 파일을 열고 access,shflag,m
　　ode에 지정된대로 공유된 읽기/쓰기를 준비한
　　다. sopen에는 세가지 플래그가 사용된다.

　　1. 읽기/쓰기 플래그

　　O_RDONLY　읽기 전용으로 개방
　　O_WRONLY　쓰기 전용으로 개방
　　O_RDWR　읽기와 쓰기용으로 개방
　　----------
　　2. 다른 억세스 플래그

　　O_NDELAY　UNIX 호환용, 여기서는 사용되지
　　않는다.
　　O_RDONLY　읽기 전용으로 개방
　　O_WRONLY　쓰기 전용으로 개방

O_RDWR 읽기와 쓰기용으로 개방
O_APPEND 이 플래그가 설정된 경우 파일 포인터가 파일끝 으로 설정된다.
O_CREAT 이 플래그는 파일이 존재할 경우 영향이 없다. 파일이 존재 하지 않을 경우 파일이 만들어 지고 mode 비트가 파일의 속성을 결정하는데 사용된다
O_TRUNC 파일이 존재하는 경우 파일의 크기는 0으로 되지만, 파일의 속성은 변경되지 않는다.
O_EXCL O_CREAT 와 같이 사용된다. 파일이 존재하는 경우 에러가 리턴된다.
O_BINARY 파일을 이진 모드로 개방한다.
O_TEXT 파일을 텍스트 모드로 개방한다.

---

3. Permission

S_IWRITE	쓰기 가능
S_IREAD	읽기 가능
S_IREAD|S_IWRITE	쓰기/읽기 가능

리 턴 값 에러가 없으면 sopen은 음이 아닌 정수(파일 handle)을 리턴한다. 에러가 있으면 -1이 리턴되며,`errno`는 다음 중 하나의 값으로 설정된다.

ENOENT Path or filename not found
EMFILE Too many open file
EACCES Permission denied
EINVAC Invalid access code

이 식 성 UNIX 시스템상에서 사용할 수 있다.
참 조 `chmod`,`close`,`creat`,`lock`,`lseek`, `_open`,`open`,`unlock`,`unmask`

쌇tell
#함수 TELL

기 능 파일 포인터의 현재 위치를 구한다.
구 문 long tell(int tell);
프로토타입 `io.h`

기능 설명 handle과 관련된 파일 포인터의 현재 위치를 구해서 파일의 선두로부터 바이트 단위로 표시한다.
리 턴 값 파일 포인터의 현재 위치를 리턴한다. 리턴값 -1은 에러를 의미하며,`errno`은 다음 값을 가진다.
이 식 성 UNIX 시스템상에서 사용할 수 있다.
참 조 `fgetpos`,`fseek`,`ftell`,`lseek`

쌇unlink
#함수 UNLINK

기 능 파일을 삭제한다.
구 문 int unlink(const char *filename);
프로토타입 `io.h`
기능 설명 filename으로 표시된 파일을 삭제한다. 어떠한 DOS 장치, 패스, 파일명도 filenaem으로 쓸 수 있다. 와일드 카드(wildcard)는 쓸 수 없다. 읽기 전용 파일은 이 함수로 삭제할 수 없다. 읽기 전용 파일을 삭제하려면 먼저 chmod나 _chmod를 호출하여 읽기 전용의 속성을 바꿔야 한다.
리 턴 값 에러가 없으면 0을 리턴한다. 에러가 발생하면 -1을 리턴하며,`errno`는 다음 중 하나로 설정된다.

ENOENT Path or filename not found
EACCES Permission denied
이 식 성 UNIX 시스템에서 사용될 수 있다.
참 조 `chmod`,`remove`

쌇unlock
#함수 UNLOCK

기 능 파일 공유에 대한 제한을 삭제한다.
구 문 int unlock(int handle, long offset, long length);
프로토타입 `io.h`
기능 설명 unlock은 이전의 lock에 의한 제한을 해제한다. 에러를 방지하기 위하여 파일을 닫기 전에 모든 제한을 해제하여야 한다. 프로그램은

종료하기 전에 모든 제한을 해제하여야 한다.
리 턴 값 에러가 없으면 0을 리턴한다. 에러가 발생하면 -1을 리턴한다.
이 식 성 DOS에 한한다.
참 조 `lock`

쌇_write
#함수 _WRITE

기 능 파일에 기록(write) 한다.
구 문 int _write(int handle, void *buf, unsigned len);
프로토타입 `io.h`
기능 설명 handle과 관련된 파일에 buf가 지정한 버퍼로부터 len만큼의 바이트를 기록한다.
_write는 모든 파일을 이진 파일로 간주한다.
실제 기록된 양이 시도된 양보다 작으면, 에러가 생긴것으로 보아야하며, 통상 디스크가 가득찬 경우이다.
리 턴 값 기록된 바이트수를 리턴한다. 에러시에는 -1을 리턴하며`errno`는 다음과 같은 값을 갖는다.

EACCES Access denied
EBADF Invalid file handle
이 식 성 DOS에 한한다.
참 조 `lseek`,`_read`,`write`

쌇write
#함수 WRITE

기 능 파일에 기록(write) 한다.
구 문 int write(int handle,void *buf, unsigned len);
프로토타입 `io.h`
handle과 관련된 파일에 buf가 지정한 버퍼로부터 len만큼의 바이트를 기록한다.
_write는 모든 파일을 이진 파일로 간주한다.
실제 기록된 양이 시도된 양보다 작으면, 에러가 생긴것으로 보아야하며, 통상 디스크가 가득찬 경우이다.

리 턴 값 기록된 바이트수를 리턴한다. 에러시에는 -1
을 리턴하며`errno`는 다음과 같은 값을 갖는
다.

EACCES Access denied
EBADF Invalid file handle
이 식 성 DOS에 한한다.
참 조 `lseek`,`_read`,`_write`
쌇limits.h
#헤더파일 LIMITS.H

각 데이타형의 최대값과 최소 값이 정의 되어있다.

CHAR_BIT SCHAR_MIN
CHAR_MAX SHRT_MAX
CHAR_MIN SHRT_MIN
INT_MAX UCHAR_MAX
INT_MIN UINT_MAX
LONG_MAX ULONG_MAX
LONG_MIN USHRT_MAX
SCHAR_MAX

쌇math.h
#헤더파일 MATH.H

함수

`abs` `acos` `asin`
`atan` `atan2` `atof`
`cabs` `ceil` `cos`
`cosh` `exp` `fabs`
`floor` `fmod` `frexp`
`hypot` `labs` `ldexp`
`log` `log10` `matherr`
`modf` `poly` `pow`
`pow10` `sin` `sinh`
`sqrt` `tan` `tanh`

쌇abs
#함수 ABS

기 능 정수의 절대값을 리턴한다.
구 문 #include <math.h>
int abs(int x);
프로토타입 `math.h`,`stdlib.h`
기능 설명 정수 인수 x의 절대값을 리턴한다. stdlib.h 가 포함되었을 때 abs는 매크로 함수로 처리 된다.
리 턴 값 인수 -32786이 -32786으로 리턴되는것을 제외 하고는 0에서 32767 사이의 정수를 리턴한다.
이 식 성 UNIX 시스템에서 사용될 수 있으며 ANSI-C 와 도 호환된다.
참 조 `cabs`,`fabs`,`labs`

쌇acos
#함수 ACOS

기 능 아크 코사인을 계산한다.
구 문 #include <math.h>
double acos(double x);
프로토타입 `math.h`
기능 설명 입력값의 아크 코사인을 리턴한다. acos에 대 한 인수는 -1 ~ 1 의 범위이다. 범위 밖의 인 수는 0을 리턴하게 한다. 이 때`errno`은 다 음과 같이 설정된다.

EDOM Domain error
리 턴 값 0 ~ pi 범위의 값을 리턴한다. 이 루틴에 대 한 에러처리 함수는 함수 matherr를 통해 수 정 된다.
이 식 성 UNIX 시스템에서 사용될 수 있으며 ANSI-C 와 도 호환된다.
참 조 `asin`,`atan`,`atan2`,`cos`,`matherr`, `sin`,`sinh`,`tan`,`tanh`

쌇asin
#함수 ASIN

기 능 아크 사인을 리턴한다.
구 문 #include <math.h>
double asin(double x);
프로토타입 `math.h`

기능 설명 입력값의 아크 사인을 리턴한다. asin에 대한 인수는 -1 ~ 1 의 범위이다. 범위 밖의 인수는 0을 리턴하게 한다. 이 때`errno`은 다음과 같이 설정된다.

EDOM Domain error

리 턴 값 -pi/2 ~ pi/2 범위의 값을 리턴한다. 이 루틴에 대한 에러처리 함수는 함수 matherr를 통해 수정 된다.

이 식 성 UNIX 시스템에서 사용될 수 있으며 ANSI-C 와도 호환된다.

참 조 `acos`,`atan`,`atan2`,`cos`,`cosh`, `matherr`,`sin`,`sinh`,`tan`,`tanh`

쌇atan

#함수 ATAN

기 능 아크 탄젠트를 계산한다.

구 문 #include <math.h>
double atan(double x);

프로토타입 `math.h`

기능 설명 입력값의 아크 탄젠트를 계산한다.

리 턴 값 -pi/2 ~ pi/2 사이의 값을 리턴한다. 이 루틴에 대한 에러처리 함수는 matherr을 통해 수정 된다.

이 식 성 UNIX 시스템에서 사용될 수 있으며 ANSI-C 와도 호환된다.

참 조 `acos`,`asin`,`atan2`,`cos`,`cosh`, `matherr`,`sin`,`sinh`,`tan`,`tanh`

쌇atan2

#함수 ATAN2

기 능 y/x의 아크 탄젠트를 계산한다.

구 문 #include <math.h>
double atan2(double y,double x);

프로토타입 `math.h`

기능 설명 y/x의 아크 탄젠트를 리턴하고, 생성된 각이 pi/2나 -pi/2에 근접해 있을 때 (x->0) 에도 정확한 결과를 산출한다.

리 턴 값 -pi ~ pi 사이의 값을 리턴한다. 이 루틴에

대한 에러처리 함수는 matherr을 통해 수정
된다.

이 식 성 UNIX 시스템에서 사용될 수 있으며 ANSI-C 와
도 호환된다.

참 조 `acos`,`asin`,`atan`,`cos`,`cosh`,
`matherr`,`sin`,`sinh`,`tan`,`tanh`

쌇atof

#함수 ATOF

기 능 문자열을 부동 소숫점으로 변환한다.

구 문 #include <math.h>
souble atof(const char *s);

프로토타입 `math.h`,`stdlib.h`

기능 설명 s에 의해 지정된 문자열을 double로 변환한다
이 함수는 부동소숫점의 문자열을 나타내는
것으로 다음과 같이 구성된다.

* 탭과 공백이 있는 임의의 문자열
* 임의의 부호
* 디지트 문자열과 임의의 소수점
* 보호정수 뒤에오는 임의의 e나 E

문자의 일반적인 형태는 다음과 같다.
[ws] [sn] [ddd] [.] [ddd] [fmt[sn]ddd]

리 턴 값 입력문자의 변환된 값을 리턴한다. 문자열이
double로 변환될 수 없다면 리턴값은 0이다.

이 식 성 UNIX 시스템에서 사용될 수 있으며 ANSI-C 와
도 호환된다.

참 조 `atoi`,`atol`,`ecvt`,`gcvt`,`strtod`

쌇cabs

#함수 CABS

기 능 복소수의 절대값

구 문 #include <math.h>
double cabs(struct complex z);

프로토타입 `math.h`

기능 설명 복소수 z의 절대값을 계산하는 매크로이다. z
는 complex형을 갖는 구조체이며 이 구조체는
다음과 같이 정의된다.

struct complex { double x, y; };

여시거 x는 실수 부분이고 y는 허수 부분이다
cabs를 호출하는것은 z의 실수와 허수 구성
부분을 갖는 sqrt를 호출하는 것과 같다.

리 턴 값 z의 절대값을 double로 리턴한다. 이 루틴에
대한 에러처리 함수는 matherr을 통해 수정
된다.

이 식 성 UNIX 시스템상에서 사용할 수 있다.

참 조 `abs`,`fabs`,`labs`,`matherr`

쌇ceil
#함수 CEIL

기 능 부동 소숫점 수의 소숫점 이하를 반올림한다.

구 문 #include <math.h>
double ceil(double x);

프로토타입 `math.h`

기능 설명 x보다 작지않은 최소의 정수를 갖는다.

리 턴 값 double 형의 정수를 리턴한다.

이 식 성 UNIX 시스템에서 사용될 수 있으며 ANSI-C 와
도 호환된다.

참 조 `floor`,`fmod`

쌇cos
#함수 COS

기 능 코사인 값을 계산한다.

구 문 #include <math.h>
double cos(double x);

프로토타입 `math.h`

기능 설명 입력된 값의 코사인 값을 리턴한다. 각도는
라디안 으로 지정된다.

리 턴 값 -1 ~ 1 사이의 범위값을 리턴한다. 이 루틴에
대한 에러처리 함수는 matherr을 통해 수정된
다.

이 식 성 UNIX 시스템에서 사용될 수 있으며 ANSI-C 와
도 호환된다.

참 조 `acos`,`sin`,`atan`,`atan2`,`cosh`,
`matherr`,`sin`,`sinh`,`tan`,`tanh`

쌇cosh
#함수 COSH

기　　능　쌍곡선 코사인을 계산한다.
구　　문　#include <math.h>
　　double cosh(double x);
프로토타입　`math.h`
기능 설명　실인수 x에 대한 쌍곡선 코사인을 계산한다.
리 턴 값　인수 x의 쌍곡선 코사인값을 나타낸다. 오버 플로우가 발생하면`errno`에 ERANGE를 설정한다. 에러처리는 matherr을 통해 변경될 수 있다.
이 식 성　UNIX 시스템에서 사용될 수 있으며 ANSI-C 와도 호환된다.
참　　조　`acos`,`asin`,`atan`,`atan2`,`cos`,`cos`, `matherr`,`sin`,`sinh`,`tan`,`tanh`

쌇exp
#함수 EXP

기　　능　지수 함수를 계산한다.
구　　문　#include <math.h>
　　double exp(double x);
프로토타입　`math.h`
기능 설명　지수함수 ex의 값을 계산한다.
리 턴 값　지수함수 ex의 값을 리턴한다. 이 루틴에 대한 에러처리 함수는 matherr을 통해 수정된다
이 식 성　UNIX 시스템에서 사용될 수 있으며 ANSI-C 와도 호환된다.
참　　조　`frexp`,`idexp`,`log`,`log10`,`matherr`, `pow`,`pow10`,`sqrt`

쌇fabs
#함수 FABS

기　　능　부동 소숫점 수의 절대값을 계산한다.
구　　문　#include <math.h>
　　double fabs(double x);
프로토타입　`math.h`
기능 설명　double 부동 소숫점 수 x의 절대값을 계산한

다.
리 턴 값 x의 절대값을 리턴하며 에러 발생시 아무것도 리턴하지 않는다.
이 식 성 UNIX 시스템에서 사용될 수 있으며 ANSI-C 와도 호환된다.
참 조 `abs`,`cabs`,`labs`

쌿floor
#함수 FLOOR

기 능 소수점 이하를 버린다.
구 문 #include <math.h>
double floor(double x);
프로토타입 `math.h`
기능 설명 x보다 크지않은 정수를 리턴한다.
리 턴 값 발견된 정수(double 형)를 리턴한다.
이 식 성 UNIX 시스템에서 사용될 수 있으며 ANSI-C 와도 호환된다.
참 조 `ceil`,`fmod`

쌿fmod
#함수 FMOD

기 능 x를 y로 나눈 나머지를 계산한다.
구 문 #include <math.h>
double fmod(double x, double y);
프로토타입 `math.h`
기능 설명 x 나누기 y를 계산한다.
리 턴 값 나머지를 리턴한다.
이 식 성 ANSI-C와 호환된다.
참 조 `ceil`,`floor`,`modf`

쌿frexp
#함수 FREXP

기 능 double형의 수치를 가수부와 지수부로 분리한다.
구 문 #include <math.h>
double frexp(double x,int *exponent);
프로토타입 `math.h`
기능 설명 가수부 m(0.5<m<1인 double형 수치)과 정수치

n을 원래 double형인 x=m*2n 을 계산한다. frexp는 n을 exponent가 가리키는 정수에 저장한다.

리 턴 값 가수부 m을 리턴한다. 이 루틴에 대한 에러처리 함수는 matherr을 통해 수정된다

이 식 성 UNIX 시스템에서 사용될 수 있으며 ANSI-C 와도 호환된다.

참 조 `exp`,`ldexp`

쌇hypot

#함수 HYPOT

기 능 직각 삼각형의 빗변의 길이를 계산한다.

구 문 #include <math.h>
double hypot(double x, double y);

프로토타입 `math.h`

기능 설명 z2=x2+y2, z>=0에서 z의 값을 계산한다. 이는 두변의 길이가 각각 x와 y인 직각 삼각형의 빗변의 길이와 같다.

리 턴 값 z를 리턴한다. 이 루틴에 대한 에러처리 함수는 matherr을 통해 수정된다

이 식 성 UNIX 시스템상에서만 사용할 수 있다.

쌇labs

#함수 LABS

기 능 long 형 절대값을 구한다.

구 문 #include <math.h>
long int labs(long int x);

프로토타입 `math.h`,`stdlib.h`

기능 설명 파라미터 x의 절대값을 구한다.

리 턴 값 에러가 없으면 x의 절대값을 리턴한다. 에러리턴값은 없다.

이 식 성 UNIX 시스템에서 사용될 수 있으며 ANSI-C 와도 호환된다.

참 조 `abs`,`cabs`,`fabs`

쌇ldexp

#함수 LDEXP

기 능 x * 2exp를 계산한다.

구　　문　#include <math.h>
　　double ldexp(double x,int exp);
프로토타입　`math.h`
기능 설명　double형 값 x * 2exp를 계산한다.
리 턴 값　에러가 없으면 ldexp는 계산한 값 x * 2exp를 리턴한다. 이 루틴에 대한 에러처리 함수는 matherr을 통해 수정된다
이 식 성　UNIX 시스템에서 사용될 수 있으며 ANSI-C 와도 호환된다.
참　　조　`exp`,`freexp`,`modf`

쌇log
#함수 LOG

기　　능　x의 자연대수를 계산한다.
구　　문　#include <math.h>
　　double log(double x);
프로토타입　`math.h`
기능 설명　x의 자연대수를 계산한다.
리 턴 값　에러가 없으면 log는 계산된 값 in(x)를 리턴한다. 이 루틴에 대한 에러처리 함수는 mathe rr을 통해 수정된다.
이 식 성　UNIX 시스템에서 사용될 수 있으며 ANSI-C 와도 호환된다.
참　　조　`exp`,`log10`,`sqrt`

쌇log10
#함수 LOG10

기　　능　상용 로그 log10(x)를 계산한다.
구　　문　#include <math.h>
　　double log10(double x);
프로토타입　`math.h`
기능 설명　상용 로그 log10(x)를 계산한다.
리 턴 값　에러가 없으면 log는 계산된 값 log10(x)를 리턴한다. 이 루틴에 대한 에러처리 함수는 `matherr`을 통해 수정된다.
이 식 성　UNIX 시스템에서 사용될 수 있으며 ANSI-C 와도 호환된다.
참　　조　`exp`,`log`

쌇matherr
#함수 MATHERR

기　　능　사용자가 변경할 수 있는 math 에러 조정기
구　　문　#include <math.h>
　　int matherr(struct eception *e);
프로토타입　`math.h`
기능 설명　math 라이브러리에 의해 발생된 에러를 조정
　　하기 위해서 _matherr루틴이 호출하는 함수이
　　다. matherr는 에러 처리 방법을 사용자가 정
　　의할 수 있는데 사용자가 직접 자신의 math
　　에러 처리 루틴을 작성해서 대체할 수 있다.

이 식 성　여러 C컴파일러상에서 사용할 수 있으나 ANSI
　　-C 와는 호환성이 없다. 메시지를 인쇄하고
　　종료하는 UNIX 형태의 matherr는 TURBO-C 디
　　스크에 있는 MATHERR.C 에서 제공된다.

쌇modf
#함수 MODF

기　　능　정수부와 소수부로 분리한다.
구　　문　#include <math.h>
　　double modf(double x,doubl *ipart);
프로토타입　`math.h`
기능 설명　double x를 두개의 부분으로 분리한다. 즉,
　　정수부와 소수부분으로 나누어 지는데 정수
　　를 ipart에 저장하고 소수부를 리턴한다.
리 턴 값　x의 소수 부분을 리턴한다.
이 식 성　UNIX 시스템에서 사용될 수 있으며 ANSI-C 와
　　도 호환된다.
참　　조　`fmod`,`ldexp`

쌇poly
#함수 POLY

기　　능　인수로부터 다항식을 만든다.
구　　문　#include <math.h>
　　double poly(double x,int degree,
　　double coeffs[]);
프로토타입　`math.h`

기능 설명 degree 차수의 계수 coeffs[0],coeffs[1]... coeffs[degree]를 갖는 변수 x의 다항식을 만든다.
리 턴 값 주어진 x에 대해 계산된 다항식의 값을 리턴한다.
이 식 성 UNIX 시스템상에서만 사용할 수 있다.

쌇pow
#함수 POW

기 능 x의 y승을 구한다.
구 문 #include <math.h>
double pow(double x, double y);
프로토타입 `math.h`
기능 설명 x의 y승을 구한다.
리 턴 값 x의 y승을 리턴한다. 이 루틴에 대한 에러처리 함수는`matherr`을 통해 수정된다.
이 식 성 UNIX 시스템에서 사용될 수 있으며 ANSI-C 와도 호환된다.
참 조 `exp`,`pow10`,`sqrt`

쌇pow10
#함수 POW10

기 능 10의 p승을 계산한다.
구 문 #include <math.h>
double pow10(int p);
프로토타입 `math.h`
기능 설명 pow10은 10p를 계산한다.
리 턴 값 에러가 없으면 pow10은 10p를 계산한 값을 리턴한다. 이 결과는 실제로 정확한 long double 형으로 계산된다. 모든 인수가 유효하지만 때로 오버플로나 언더플로가 발생할 수 있다.
이 식 성 UNIX 시스템상에서만 사용할 수 있다.
참 조 `exp`,`pow`

쌇sin
#함수 SIN

기 능 사인값을 계산한다.
구 문 #include <math.h>

double sin(double x);
프로토타입 `math.h`
기능 설명 각도 X의 사인값을 계산한다. 각도는 라디안으로 주어진다. 이루틴의 에러 처리는 matherr 함수를 통하여 수정될 수 있다.
리 턴 값 입력 x의 사인값을 리턴한다.
이 식 성 UNIX 시스템에서 사용될 수 있으며 ANSI-C 와도 호환된다.
참 조 `acos`,`asin`,`cos`,`cosh`,`tan`,`tanh`

쌇sinh
#함수 SINH

기 능 쌍곡선 (hyperbolic) 사인값을 계산한다.
구 문 #include <math.h>
double sinh(double x);
프로토타입 `math.h`
기능 설명 실제 인수에 대한 쌍곡선 사인값을 계산한다. sinh에 대한 에러처리는 matherr을 통하여 수정될 수 있다.
리 턴 값 x의 쌍곡선 사인값을 리턴한다.
이 식 성 UNIX 시스템에서 사용될 수 있으며 ANSI-C 와도 호환된다.
참 조 `acos`,`asin`,`atan`,`atan2`,`cos`,`cosh`,`sin`,`tan`,`tanh`

쌇sqrt
#함수 SQRT

기 능 입력값의 평방근을 구한다.
구 문 #include <math.h>
double sqrt(double x);
프로토타입 `math.h`
기능 설명 입력된 값의 평방근을 계산한다. 이 루틴의 에러 처리는`matherr`을 통해 수정될 수 있다
리 턴 값 올바로 계산되었으면 sqrt는 양의 정수를 리턴한다.
이 식 성 UNIX 시스템에서 사용될 수 있으며 ANSI-C 와도 호환된다.
참 조 `exp`,`log`,`pow`

쌇tan
#함수 TAN

기　　능　탄젠트 값을 계산한다.
구　　문　#include <math.h>
　　double tan(double x);
프로토타입　`math.h`
기능 설명　탄젠트 값을 계산한다. 각도는 라디안으로 표
　　시된다. 이 루틴의 에러처리는 matherr로 수
　　정될 수 있다.
리 턴 값　x의 탄젠트 값을 리턴한다.
이 식 성　UNIX 시스템에서 사용될 수 있으며 ANSI-C 와
　　도 호환된다.
참　　조　`acos`,`asin`,`atan`,`atan2`,`cos`,`cosh`,
　　`sinh`,`tanh`

쌇tanh
#함수 TANH

기　　능　쌍곡선 탄젠트 값을 계산한다.
구　　문　#include <math.h>
　　double tanh(double x);
프로토타입　`math.h`
기능 설명　실인수의 쌍곡선 탄젠트 값을 리턴한다. 이
　　루틴의 에러처리는`matherr`을 통해서 수정될
　　수 있다.
리 턴 값　x의 쌍곡선 탄젠트 값을 리턴한다.
이 식 성　UNIX 시스템에서 사용될 수 있으며 ANSI-C 와
　　도 호환된다.
참　　조　`acos`,`asin`,`atan`,`atan2`,`cos`,`cosh`,
　　`sin`,`sinh`,`tan`

쌇mem.h
#헤더파일 MEM.H

함수

`memccpy`　`memchr`　`memcmp`
`memcpy`　`memicmp`　`memmove`
`memset`　`movedata`　`movmem`
`setmem`

쌇memccpy
#함수 MEMCCPY

기　　능　n바이트의 블록을 복사한다.
구　　문　#include <mem.h>
　　void *memccpy(void *dest,const void *src,
　　int c, size_t n);
프로토타입　`mem.h`
기능 설명　src로 부터 dest로 n바이트를 복사한다. 복사 중 다음과 같은 사항이 발생되면 즉시 중단된다.

　　* 문자 c가 최초로 dest에 복사됬을때
　　* n바이트를 dest에 복사했을때

리 턴 값　문자 c를 복사한 경우에 dest가 가리키고 있는 바이트에 대한 포인터를 리턴한다. 그 외의 경우에는`null`이 리턴된다.
이 식 성　UNIX 시스템 V에서 사용할 수 있다.
참　　조　`memcpy`,`memmove`,`memset`

쌇memchr
#함수 MEMCHR

기　　능　문자 c에 대한 n바이트를 검색한다.
구　　문　#include <mem.h>
　　void *memchr(const void *s, int c,
　　size_t n);
프로토타입　`string.h`,`mem.h`
기능 설명　s가 지정한 n바이트 영역에 있는 문자 c를 탐색하는 함수이다.
리 턴 값　에러가 없으면 memchr은 s에서 문자 c를 발견한 최초의 문자 c에대한 포인터를 리턴한다. 문자 c를 발견하지 못한 경우에는 null을 리턴 한다.
이 식 성　UNIX 시스템 V에서 사용할 수 있으며 ANSI-C와도 호환된다.

쌇memcmp
#함수 MEMCMP

기 능 정확하게 n바이트 길이에 대한 두 개의 블록
을 비교한다.
구 문 #include <mem.h>
int memcmp(const void *s1, const void *s2,
size_t n);
프로토타입 `string.h`,`mem.h`
기능 설명 블록 s1과 s2의 초기 n바이트 블록을 비교한
다
리 턴 값 다음값을 리턴한다.

<0 (s1이 s2보다 작을경우)
=0 (s1과 s2이 같을 경우)
>0 (s1이 s2보다 클 경우)
이 식 성 UNIX 시스템 V에서 사용할 수 있으며 ANSI-C
와도 호환된다.
참 조 `memicmp`

쌇memcpy
#함수 MEMCPY

기 능 n바이트의 블록을 복사한다.
구 문 #include <mem.h>
void *memcpy(void *dest,void *src,
size_t n);
프로토타입 `string.h`,`mem.h`
기능 설명 n바이트의 블록을 src에서 dest로 복사한다.
src와 dest가 중복되는 경우 결과를 예상할
수 없다.
리 턴 값 dest를 리턴한다.
이 식 성 UNIX 시스템 V에서 사용할 수 있으며 ANSI-C
와도 호환된다.
참 조 `memccpy`,`memmove`,`memset`,`movedata`,
`movemem`

쌇memicmp
#함수 MEMICMP

기 능 두 문자 배열의 n바이트를 비교하나 활자체(
대문자와 소문자)는 무시한다. (메모리 내의
문자열 비교함수)
구 문 #include <mem.h>

int memicmp(const void *s1,const void *s2,
size_t n);
프로토타입 `string.h`,`mem.h`
기능 설명 블록 s1과 s2의 초기의 n바이트를 비교하나
소문자와 대문자는 무시한다.
리 턴 값 다음값을 리턴한다.

<0 (s1이 s2보다 작을경우)
=0 (s1과 s2이 같을 경우)
>0 (s1이 s2보다 클 경우)

이 식 성 UNIX 시스템 V에서 사용할 수 있다.
참 조 `memcmp`

쌿memmove
#함수 MEMMOVE

기 능 n바이트의 블록을 복사한다.
구 문 #include <mem.h>
void *memmove(void *dest,const void *src,
size_t n);
프로토타입 `string.h`,`mem.h`
기능 설명 src로부터 n바이브의 블록을 dest로 복사한다
src와 dest가 중복되어도 내용이 정확하게 복
사 된다.
리 턴 값 dest를 리턴한다.
이 식 성 UNIX 시스템 V에서 사용할 수 있으며 ANSI-C
와도 호환된다.
참 조 `memccpy`,`memcpy`,`movmem`

쌿memset
#함수 MEMSET

기 능 메모리의 n바이트 블록을 모두 c로 설정한다.
구 문 #include <mem.h>
void memset(void *s,int c,size_t n);
프로토타입 `mem.h`
기능 설명 배열 s의 초기 n바이트를 c로 설정한다.
리 턴 값 s를 리턴한다.
이 식 성 UNIX 시스템 V에서 사용할 수 있으며 ANSI-C
와도 호환된다.

참　　조　`memccpy`,`memcpy`,`setmem`

쌇movedata
#함수 MOVEDATA

기　　능　n바이트를 복사한다.
구　　문　void movedata(unsigned srcseg,
　　unsigned srcoff,unsigned destseg,
　　unsigned destoff,size_t n);
프로토타입　`mem.h`,`string.h`
기능 설명　소스 어드레스(scrseg:scroff)에서 n바이트를
　　목적 어드레스(destseg:destoff)로 이동한다.
　　movedata는 메모리 모델에 관계없이 데이타를
　　이동하는 방법이다.
리 턴 값　없음
참　　조　`FP_OFF`,`memcpy`,`MK_FP`,`movmem`,`segread`

쌇movmem
#함수 MOVMEM

기　　능　length바이트의 블록을 복사한다.
구　　문　void movmem(void *scr, void *dest,
　　unsinged length);
프로토타입　`mem.h`
기능 설명　scr로부터 length바이트의 블록을 dest로 복
　　사한다. 소스와 목적지가 중복되어도 복사는
　　정확하게 이루어진다.
리 턴 값　없음
참　　조　`memcpy`,`memmove`,`movedata`

쌇setmem
#함수 SETMEM

기　　능　메모리 영역에 값을 할당한다.
구　　문　void setmem(void *dest, unsigned length,
　　char value);
프로토타입　`mem.h`
기능 설명　dest에 의해 지정된 length만큼의 블록을 val
　　ue바이트로 설정한다.
리 턴 값　없음
이 식 성　8086 계열에 한한다.

참 조 `memset`,`strest`

쌇process.h
#헤더파일 PROCESS.H

함수

`abort` `execl` `execle`
`execlp` `execlpe` `execv`
`execve` `execvp` `execvpe`
`_exit` `exit` `spawnl`
`spawnle` `spawnlp` `spawnlpe`
`spawnv` `spawnve` `spawnvp`
`spawnvpe` `system`

쌇abort
#함수 ABORT

기 능 수행중인 프로세스를 변칙적으로 종료시킨다.
구 문 void abort(void);
프로토타입 `stdlib.h`,`process.h`
기능 설명 stderr에 프로세스 종료 메시지를(수행중인 프로그램의 종료)를 출력하고 exit code가 3인 _exit 함수를 호출하여 프로그램을 종료하는 함수이다.
리 턴 값 exit code 3을 현재의 프로세스나 DOS에 리턴한다.
이 식 성 UNIX 시스템에서 사용될 수 있으며 ANSI-C 와도 호환된다.
참 조 `assert`,`atexit`,`exit`,`_exit`,`raise`, `signal`

쌇exec
#함수 EXEC

기 능 다른 프로그램을 로드한후 실행한다.
구 문 int execl(char *path,char *arg0,*arg1,.... NULL);
int execle(char *path,char *arg0,*arg1,.... NULL,char **env);
int execlp(char *path,char *arg0,*argn, NULL);

int execlpe(char *path,char *arg0,*arg1,....
NULL,char **env);
int execv(char *path,char *argv[]);
int execve(char *path,char *argv[],
char **env);
int execvp(char *path,char *argv[]);
int execvpe(char *path,char *argv[],
char **env);

프로토타입 `process.h`

기능 설명 exec.. 계의 함수들은 child peocesses로 알려진 다른 프로그램을 로드하고 실행한다.
exec.. 호출이 성공적일때 생성 프로세스는 / 모 프로세스를 오버레이 해서 파괴시킨다.
exec.. 뒤에 접미사 l, v, p, e 등이 붙어서 각각의 함수명을 이루며 어떤 특정한 기능을 가지고 작동한다.

p: DOS PATH 환경변수가 지정하는 디렉토리에서 실행할 파일을 탐색한다.
l: 인수 포인터 (arg0, arg1, ...,argn)가 분리된 인수로서 전달되는데 보통 전달될 인수의 수를 미리 알고있을때 사용한다.
v: 인수 포인터 (argv[0]..argv[n])가 포인터의 배열로 전달된다. 인수의 수가 가변적일때 사용된다.
e: 인수 env가 생성 프로세스에 전달되어 생성 프로세스의 화경변수를 변경할 수 있게 한다. 접미사 e가 없으면 생성 프로세스는 모 프로세스의 환경을 물려받게 된다.

리 턴 값 생성 프로세스가 성공적으로 되면 exec.. 함수는 아무값도 리턴하지 않고 에러가 생긴 경우에는 -1을 리턴한다.

이 식 성 DOS에 한한다.

참 조 `abort`,`atexit`,`_exit`,`exit`,`_fpreset`, `searchpath`,`spawn`,`system`

쌇execl
#함수 EXECL

기 능 다른 프로그램을 로드한후 실행한다.

구 문 int execl(char *path,char *arg0,*arg1,....
NULL);
int execle(char *path,char *arg0,*arg1,....
NULL,char **env);
int execlp(char *path,char *arg0,*argn,
NULL);
int execlpe(char *path,char *arg0,*arg1,....
NULL,char **env);
int execv(char *path,char *argv[]);
int execve(char *path,char *argv[],
char **env);
int execvp(char *path,char *argv[]);
int execvpe(char *path,char *argv[],
char **env);

프로토타입 `process.h`

기능 설명 exec.. 계의 함수들은 child peocesses로 알려진 다른 프로그램을 로드하고 실행한다.
exec.. 호출이 성공적일때 생성 프로세스는
/ 모 프로세스를 오버레이 해서 파괴시킨다.
exec.. 뒤에 접미사 l, v, p, e 등이 붙어서 각각의 함수명을 이루며 어떤 특정한 기능을 가지고 작동한다.

p: DOS PATH 환경변수가 지정하는 디렉토리에서 실행할 파일을 탐색한다.
l: 인수 포인터 (arg0, arg1, ...,argn)가 분리된 인수로서 전달되는데 보통 전달될 인수의 수를 미리 알고있을때 사용한다.
v: 인수 포인터 (argv[0]..argv[n])가 포인터의 배열로 전달된다. 인수의 수가 가변적일때 사용된다.
e: 인수 env가 생성 프로세스에 전달되어 생성 프로세스의 화경변수를 변경할 수 있게 한다. 접미사 e가 없으면 생성 프로세스는 모 프로세스의 환경을 물려받게 된다.

리 턴 값 생성 프로세스가 성공적으로 되면 exec.. 함수는 아무값도 리턴하지 않고 에러가 생긴 경우에는 -1을 리턴한다.

이 식 성 DOS에 한한다.
참 조 `abort`,`atexit`,`_exit`,`exit`,`_fpreset`,
`searchpath`,`spawn`,`system`

쌇execle
#함수 EXECLE

기 능 다른 프로그램을 로드한후 실행한다.
구 문 int execl(char *path,char *arg0,*arg1,....
NULL);
int execle(char *path,char *arg0,*arg1,....
NULL,char **env);
int execlp(char *path,char *arg0,*argn,
NULL);
int execlpe(char *path,char *arg0,*arg1,....
NULL,char **env);
int execv(char *path,char *argv[]);
int execve(char *path,char *argv[],
char **env);
int execvp(char *path,char *argv[]);
int execvpe(char *path,char *argv[],
char **env);
프로토타입 `process.h`
기능 설명 exec.. 계의 함수들은 child peocesses로 알
려진 다른 프로그램을 로드하고 실행한다.
exec.. 호출이 성공적일때 생성 프로세스는
/ 모 프로세스를 오버레이 해서 파괴시킨다.
exec.. 뒤에 접미사 l, v, p, e 등이 붙어서
각각의 함수명을 이루며 어떤 특정한 기능을
가지고 작동한다.

p: DOS PATH 환경변수가 지정하는 디렉토리에
서 실행할 파일을 탐색한다.
l: 인수 포인터 (arg0, arg1, ...,argn)가 분
리된 인수로서 전달되는데 보통 전달될 인
수의 수를 미리 알고있을때 사용한다.
v: 인수 포인터 (argv[0]..argv[n])가 포인터
의 배열로 전달된다. 인수의 수가 가변적
일때 사용된다.
e: 인수 env가 생성 프로세스에 전달되어 생
성 프로세스의 화경변수를 변경할 수 있게

한다. 접미사 e가 없으면 생성 프로세스는
모 프로세스의 환경을 물려받게 된다.

리 턴 값 생성 프로세스가 성공적으로 되면 exec.. 함
수는 아무값도 리턴하지 않고 에러가 생긴 경
우에는 -1을 리턴한다.

이 식 성 DOS에 한한다.
참 조 `abort`,`atexit`,`_exit`,`exit`,`_fpreset`,
`searchpath`,`spawn`,`system`

쌿execlp
#함수 EXECLP
기 능 다른 프로그램을 로드한후 실행한다.
구 문 int execl(char *path,char *arg0,*arg1,....
NULL);
int execle(char *path,char *arg0,*arg1,....
NULL,char **env);
int execlp(char *path,char *arg0,*argn,
NULL);
int execlpe(char *path,char *arg0,*arg1,....
NULL,char **env);
int execv(char *path,char *argv[]);
int execve(char *path,char *argv[],
char **env);
int execvp(char *path,char *argv[]);
int execvpe(char *path,char *argv[],
char **env);
프로토타입 `process.h`
기능 설명 exec.. 계의 함수들은 child peocesses로 알
려진 다른 프로그램을 로드하고 실행한다.
exec.. 호출이 성공적일때 생성 프로세스는
/ 모 프로세스를 오버레이 해서 파괴시킨다.
exec.. 뒤에 접미사 l, v, p, e 등이 붙어서
각각의 함수명을 이루며 어떤 특정한 기능을
가지고 작동한다.

p: DOS PATH 환경변수가 지정하는 디렉토리에
서 실행할 파일을 탐색한다.
l: 인수 포인터 (arg0, arg1, ...,argn)가 분
리된 인수로서 전달되는데 보통 전달될 인

수의 수를 미리 알고있을때 사용한다.
v: 인수 포인터 (argv[0]..argv[n])가 포인터의 배열로 전달된다. 인수의 수가 가변적일때 사용된다.
e: 인수 env가 생성 프로세스에 전달되어 생성 프로세스의 화경변수를 변경할 수 있게 한다. 접미사 e가 없으면 생성 프로세스는 모 프로세스의 환경을 물려받게 된다.

리 턴 값 생성 프로세스가 성공적으로 되면 exec.. 함수는 아무값도 리턴하지 않고 에러가 생긴 경우에는 -1을 리턴한다.

이 식 성 DOS에 한한다.
참 조 `abort`,`atexit`,`_exit`,`exit`,`_fpreset`, `searchpath`,`spawn`,`system`

쌿execlpe
#함수 EXECLPE
기 능 다른 프로그램을 로드한후 실행한다.
구 문 int execl(char *path,char *arg0,*arg1,.... NULL);
int execle(char *path,char *arg0,*arg1,.... NULL,char **env);
int execlp(char *path,char *arg0,*argn, NULL);
int execlpe(char *path,char *arg0,*arg1,.... NULL,char **env);
int execv(char *path,char *argv[]);
int execve(char *path,char *argv[], char **env);
int execvp(char *path,char *argv[]);
int execvpe(char *path,char *argv[], char **env);
프로토타입 `process.h`
기능 설명 exec.. 계의 함수들은 child peocesses로 알려진 다른 프로그램을 로드하고 실행한다.
exec.. 호출이 성공적일때 생성 프로세스는 / 모 프로세스를 오버레이 해서 파괴시킨다.
exec.. 뒤에 접미사 l, v, p, e 등이 붙어서 각각의 함수명을 이루며 어떤 특정한 기능을

가지고 작동한다.

p: DOS PATH 환경변수가 지정하는 디렉토리에서 실행할 파일을 탐색한다.
l: 인수 포인터 (arg0, arg1, ...,argn)가 분리된 인수로서 전달되는데 보통 전달될 인수의 수를 미리 알고있을때 사용한다.
v: 인수 포인터 (argv[0]..argv[n])가 포인터의 배열로 전달된다. 인수의 수가 가변적일때 사용된다.
e: 인수 env가 생성 프로세스에 전달되어 생성 프로세스의 화경변수를 변경할 수 있게 한다. 접미사 e가 없으면 생성 프로세스는 모 프로세스의 환경을 물려받게 된다.

리 턴 값 생성 프로세스가 성공적으로 되면 exec.. 함수는 아무값도 리턴하지 않고 에러가 생긴 경우에는 -1을 리턴한다.

이 식 성 DOS에 한한다.
참 조 `abort`,`atexit`,`_exit`,`exit`,`_fpreset`, `searchpath`,`spawn`,`system`

쌇execv
#함수 EXECV
기 능 다른 프로그램을 로드한후 실행한다.
구 문 int execl(char *path,char *arg0,*arg1,.... NULL);
int execle(char *path,char *arg0,*arg1,.... NULL,char **env);
int execlp(char *path,char *arg0,*argn, NULL);
int execlpe(char *path,char *arg0,*arg1,.... NULL,char **env);
int execv(char *path,char *argv[]);
int execve(char *path,char *argv[], char **env);
int execvp(char *path,char *argv[]);
int execvpe(char *path,char *argv[], char **env);
프로토타입 `process.h`

기능 설명 exec.. 계의 함수들은 child peocesses로 알
려진 다른 프로그램을 로드하고 실행한다.
exec.. 호출이 성공적일때 생성 프로세스는
/ 모 프로세스를 오버레이 해서 파괴시킨다.
exec.. 뒤에 접미사 l, v, p, e 등이 붙어서
각각의 함수명을 이루며 어떤 특정한 기능을
가지고 작동한다.

p: DOS PATH 환경변수가 지정하는 디렉토리에
서 실행할 파일을 탐색한다.
l: 인수 포인터 (arg0, arg1, ...,argn)가 분
리된 인수로서 전달되는데 보통 전달될 인
수의 수를 미리 알고있을때 사용한다.
v: 인수 포인터 (argv[0]..argv[n])가 포인터
의 배열로 전달된다. 인수의 수가 가변적
일때 사용된다.
e: 인수 env가 생성 프로세스에 전달되어 생
성 프로세스의 화경변수를 변경할 수 있게
한다. 접미사 e가 없으면 생성 프로세스는
모 프로세스의 환경을 물려받게 된다.

리 턴 값 생성 프로세스가 성공적으로 되면 exec.. 함
수는 아무값도 리턴하지 않고 에러가 생긴 경
우에는 -1을 리턴한다.

이 식 성 DOS에 한한다.
참 조 `abort`,`atexit`,`_exit`,`exit`,`_fpreset`,
`searchpath`,`spawn`,`system`

쌇execve
#함수 EXEVE
기 능 다른 프로그램을 로드한후 실행한다.
구 문 int execl(char *path,char *arg0,*arg1,....
NULL);
int execle(char *path,char *arg0,*arg1,....
NULL,char **env);
int execlp(char *path,char *arg0,*argn,
NULL);
int execlpe(char *path,char *arg0,*arg1,....
NULL,char **env);
int execv(char *path,char *argv[]);

int execve(char *path,char *argv[],
char **env);
int execvp(char *path,char *argv[]);
int execvpe(char *path,char *argv[],
char **env);

프로토타입 `process.h`

기능 설명 exec.. 계의 함수들은 child peocesses로 알려진 다른 프로그램을 로드하고 실행한다. exec.. 호출이 성공적일때 생성 프로세스는 / 모 프로세스를 오버레이 해서 파괴시킨다. exec.. 뒤에 접미사 l, v, p, e 등이 붙어서 각각의 함수명을 이루며 어떤 특정한 기능을 가지고 작동한다.

p: DOS PATH 환경변수가 지정하는 디렉토리에서 실행할 파일을 탐색한다.

l: 인수 포인터 (arg0, arg1, ...,argn)가 분리된 인수로서 전달되는데 보통 전달될 인수의 수를 미리 알고있을때 사용한다.

v: 인수 포인터 (argv[0]..argv[n])가 포인터의 배열로 전달된다. 인수의 수가 가변적일때 사용된다.

e: 인수 env가 생성 프로세스에 전달되어 생성 프로세스의 화경변수를 변경할 수 있게 한다. 접미사 e가 없으면 생성 프로세스는 모 프로세스의 환경을 물려받게 된다.

리 턴 값 생성 프로세스가 성공적으로 되면 exec.. 함수는 아무값도 리턴하지 않고 에러가 생긴 경우에는 -1을 리턴한다.

이 식 성 DOS에 한한다.

참 조 `abort`,`atexit`,`_exit`,`exit`,`_fpreset`, `searchpath`,`spawn`,`system`

쌇execvp

#함수 EXECVP

기 능 다른 프로그램을 로드한후 실행한다.

구 문 int execl(char *path,char *arg0,*arg1,.... NULL);
int execle(char *path,char *arg0,*arg1,....

NULL,char **env);
int execlp(char *path,char *arg0,*argn,
NULL);
int execlpe(char *path,char *arg0,*arg1,....
NULL,char **env);
int execv(char *path,char *argv[]);
int execve(char *path,char *argv[],
char **env);
int execvp(char *path,char *argv[]);
int execvpe(char *path,char *argv[],
char **env);

프로토타입 `process.h`

기능 설명 exec.. 계의 함수들은 child peocesses로 알려진 다른 프로그램을 로드하고 실행한다. exec.. 호출이 성공적일때 생성 프로세스는 / 모 프로세스를 오버레이 해서 파괴시킨다. exec.. 뒤에 접미사 l, v, p, e 등이 붙어서 각각의 함수명을 이루며 어떤 특정한 기능을 가지고 작동한다.

p: DOS PATH 환경변수가 지정하는 디렉토리에서 실행할 파일을 탐색한다.
l: 인수 포인터 (arg0, arg1, ...,argn)가 분리된 인수로서 전달되는데 보통 전달될 인수의 수를 미리 알고있을때 사용한다.
v: 인수 포인터 (argv[0]..argv[n])가 포인터의 배열로 전달된다. 인수의 수가 가변적일때 사용된다.
e: 인수 env가 생성 프로세스에 전달되어 생성 프로세스의 화경변수를 변경할 수 있게 한다. 접미사 e가 없으면 생성 프로세스는 모 프로세스의 환경을 물려받게 된다.

리 턴 값 생성 프로세스가 성공적으로 되면 exec.. 함수는 아무값도 리턴하지 않고 에러가 생긴 경우에는 -1을 리턴한다.

이 식 성 DOS에 한한다.

참 조 `abort`,`atexit`,`_exit`,`exit`,`_fpreset`, `searchpath`,`spawn`,`system`

쌇execvpe

#함수 EXEVPE

기 능 다른 프로그램을 로드한후 실행한다.

구 문 int execl(char *path,char *arg0,*arg1,....
NULL);
int execle(char *path,char *arg0,*arg1,....
NULL,char **env);
int execlp(char *path,char *arg0,*argn,
NULL);
int execlpe(char *path,char *arg0,*arg1,....
NULL,char **env);
int execv(char *path,char *argv[]);
int execve(char *path,char *argv[],
char **env);
int execvp(char *path,char *argv[]);
int execvpe(char *path,char *argv[],
char **env);

프로토타입 `process.h`

기능 설명 exec.. 계의 함수들은 child peocesses로 알려진 다른 프로그램을 로드하고 실행한다.
exec.. 호출이 성공적일때 생성 프로세스는
/ 모 프로세스를 오버레이 해서 파괴시킨다.
exec.. 뒤에 접미사 l, v, p, e 등이 붙어서 각각의 함수명을 이루며 어떤 특정한 기능을 가지고 작동한다.

p: DOS PATH 환경변수가 지정하는 디렉토리에서 실행할 파일을 탐색한다.
l: 인수 포인터 (arg0, arg1, ...,argn)가 분리된 인수로서 전달되는데 보통 전달될 인수의 수를 미리 알고있을때 사용한다.
v: 인수 포인터 (argv[0]..argv[n])가 포인터의 배열로 전달된다. 인수의 수가 가변적일때 사용된다.
e: 인수 env가 생성 프로세스에 전달되어 생성 프로세스의 화경변수를 변경할 수 있게 한다. 접미사 e가 없으면 생성 프로세스는 모 프로세스의 환경을 물려받게 된다.

리 턴 값 생성 프로세스가 성공적으로 되면 exec.. 함수는 아무값도 리턴하지 않고 에러가 생긴 경

우에는 -1을 리턴한다.

이 식 성 DOS에 한한다.
참 조 `abort`,`atexit`,`_exit`,`exit`,`_fpreset`,
`searchpath`,`spawn`,`system`

쌇_exit
#함수 _EXIT

기 능 프로그램의 실행을 종료한다.
구 문 void _exit(int status);
프로토타입 `process.h`,`stdlib.h`
기능 설명 _exit는 프로그램의 실행이 종료될 때에 개방
된 파일을 닫지 않으며, 출력이나 어떠한 종
료 함수 호출도 하지 않는다. status는 프로
세스의 종료 상태로서 호출 프로세스에 부여
되며 일반적으로 정상적인 종료를 나타내기
위해서는 0이 사용되고 0이외의 값이 사용되
면 에러가 있다는 것을 나타낸다.
리 턴 값 없음
이 식 성 UNIX 시스템상에서 사용할 수 있다.
참 조 `abort`,`atexit`,`exec`,`exit`,`spawn`

쌇exit
#함수 EXIT

기 능 프로그램의 실행을 종료한다.
구 문 void exit(int status);
프로토타입 `process.h`,`stdlib.h`
기능 설명 호출하는 프로세스를 종료할 경우 종료전의
모든 파일을 닫고 버퍼에 output(출력 대기중
)이 작성되며 함수 atexit로 등록된 "종료함
수"(exit function)가 호출된다.
리 턴 값 없음
이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C 와
도 호환된다.
참 조 `abort`,`atexit`,`exec`,`_exit`,`spawn`

쌇spawn
#함수 SPAWN

기　　능　생성 (child) 프로세스를 만들고 실행시킨다.
구　　문

```
#include <process.h>
#include <stdio.h>
int spawn(int mode, char *path, char *arg0,*arg1,...,
    argn,null);
int spawnle(int mode, char *path, char *arg0,
    *arg1,...,argn, null, char *envp[]);
int spawnlp(int mode, char *path, char *arg0,
    *arg1,...,argn,null);
int spawnlpe(int mode, char *path, char *arg0,
    *arg1,...,argn,null,char *envp[]);
int spawnv(int mode, char *path, char *argv[],
    char argp[]);
int spawnvp(int mode, char *path, char *argv[]);
int spawnvpe(int mode, char *path, char *argv[],
    char *argp[]);
```

프로토타입　`process.h`
기능 설명　spawn..에 속하는 함수들은 생성 프로세스(ch
ild process)로 알려진 다른 파일을 메모리에
올리고 실행시킨다.
이 때 생성 프로세스를 위한 충분한 메모리
공간이 있어야 한다.
mode 값은 실행시키는 모 프로세스가 생성 프
로세스를 개시한 뒤에 어떤 작동을 하는지를
결정한다. mode 값은 아래와 같다.

P_WAIT　생성 프로세스가 실행을 끝낼 때까
지 모프로세스의 실행을 중지한다.
P_NOWAIT　생성 프로세스를 샐행하면서 모 프
로세스도 계속 실행한다.
P_OVERAY　생성 프로세스가 모 프로세스를 겹
치도록 한다.`exec`을 호출하는 것
과 같다.
주의 P_NOWAIT 는 아직 사용할 수 없다.

spawn.. 뒤에 접미사 l, v, p, e 등이 붙어서
각각의 함수명을 이루며 어떤 특정한 기능을
가지고 작동한다.

p: DOS PATH 환경변수가 지정하는 디렉토리에서 실행할 파일을 탐색한다.

l: 인수 포인터 (arg0, arg1, ...,argn)가 분리된 인수로서 전달되는데 보통 전달될 인수의 수를 미리 알고있을때 사용한다.

v: 인수 포인터 (argv[0]..argv[n])가 포인터의 배열로 전달된다. 인수의 수가 가변적일때 사용된다.

e: 인수 env가 생성 프로세스에 전달되어 생성 프로세스의 화경변수를 변경할 수 있게 한다. 접미사 e가 없으면 생성 프로세스는 모 프로세스의 환경을 물려받게 된다.

리 턴 값 에러가 없으면 spawn 함수는 생성 프로세스의 exit 상태를 리턴한다.

참 조 `abort`,`atexit`,`_exit`,`exec`,`exit`, `_fpreset`,`searchpath`,`system`

쌿spawnl

#함수 SPAWNL

기 능 생성 (child) 프로세스를 만들고 실행시킨다.

구 문

```
#include <process.h>
#include <stdio.h>
int spawn(int mode, char *path, char *arg0,*arg1,...,
    argn,null);
int spawnle(int mode, char *path, char *arg0,
    *arg1,...,argn, null, char *envp[]);
int spawnlp(int mode, char *path, char *arg0,
    *arg1,...,argn,null);
int spawnlpe(int mode, char *path, char *arg0,
    *arg1,...,argn,null,char *envp[]);
int spawnv(int mode, char *path, char *argv[],
    char argp[]);
int spawnvp(int mode, char *path, char *argv[]);
int spawnvpe(int mode, char *path, char *argv[],
    char *argp[]);
```

프로토타입 `process.h`

기능 설명 spawn..에 속하는 함수들은 생성 프로세스(child process)로 알려진 다른 파일을 메모리에 올리고 실행시킨다.
이 때 생성 프로세스를 위한 충분한 메모리 공간이 있어야 한다.
mode 값은 실행시키는 모 프로세스가 생성 프로세스를 개시한 뒤에 어떤 작동을 하는지를 결정한다. mode 값은 아래와 같다.

P_WAIT 생성 프로세스가 실행을 끝낼 때까지 모프로세스의 실행을 중지한다.
P_NOWAIT 생성 프로세스를 샐행하면서 모 프로세스도 계속 실행한다.
P_OVERAY 생성 프로세스가 모 프로세스를 겹치도록 한다.`exec`을 호출하는 것과 같다.
주의 P_NOWAIT 는 아직 사용할 수 없다.

spawn.. 뒤에 접미사 l, v, p, e 등이 붙어서 각각의 함수명을 이루며 어떤 특정한 기능을 가지고 작동한다.

p: DOS PATH 환경변수가 지정하는 디렉토리에서 실행할 파일을 탐색한다.
l: 인수 포인터 (arg0, arg1, ...,argn)가 분리된 인수로서 전달되는데 보통 전달될 인수의 수를 미리 알고있을때 사용한다.
v: 인수 포인터 (argv[0]..argv[n])가 포인터의 배열로 전달된다. 인수의 수가 가변적일때 사용된다.
e: 인수 env가 생성 프로세스에 전달되어 생성 프로세스의 화경변수를 변경할 수 있게 한다. 접미사 e가 없으면 생성 프로세스는 모 프로세스의 환경을 물려받게 된다.

리 턴 값 에러가 없으면 spawn 함수는 생성 프로세스의 exit 상태를 리턴한다.

참 조 `abort`,`atexit`,`_exit`,`exec`,`exit`, `_fpreset`,`searchpath`,`system`

쌇spawnle
#함수 SPAWNLE

기　　능　생성 (child) 프로세스를 만들고 실행시킨다.
구　　문

```
#include <process.h>
#include <stdio.h>
int spawn(int mode, char *path, char *arg0,*arg1,...,
    argn,null);
int spawnle(int mode, char *path, char *arg0,
    *arg1,...,argn, null, char *envp[]);
int spawnlp(int mode, char *path, char *arg0,
    *arg1,...,argn,null);
int spawnlpe(int mode, char *path, char *arg0,
    *arg1,...,argn,null,char *envp[]);
int spawnv(int mode, char *path, char *argv[],
    char argp[]);
int spawnvp(int mode, char *path, char *argv[]);
int spawnvpe(int mode, char *path, char *argv[],
    char *argp[]);
```

프로토타입　`process.h`
기능　설명　spawn..에 속하는 함수들은 생성 프로세스(ch
ild process)로 알려진 다른 파일을 메모리에
올리고 실행시킨다.
이 때 생성 프로세스를 위한 충분한 메모리
공간이 있어야 한다.
mode 값은 실행시키는 모 프로세스가 생성 프
로세스를 개시한 뒤에 어떤 작동을 하는지를
결정한다. mode 값은 아래와 같다.

P_WAIT　　생성 프로세스가 실행을 끝낼 때까
지 모프로세스의 실행을 중지한다.
P_NOWAIT　생성 프로세스를 샐행하면서 모 프
로세스도 계속 실행한다.
P_OVERAY　생성 프로세스가 모 프로세스를 겹
치도록 한다.`exec`을 호출하는 것
과 같다.
주의 P_NOWAIT 는 아직 사용할 수 없다.

spawn.. 뒤에 접미사 l, v, p, e 등이 붙어서
각각의 함수명을 이루며 어떤 특정한 기능을
가지고 작동한다.

p: DOS PATH 환경변수가 지정하는 디렉토리에
서 실행할 파일을 탐색한다.
l: 인수 포인터 (arg0, arg1, ...,argn)가 분
리된 인수로서 전달되는데 보통 전달될 인
수의 수를 미리 알고있을때 사용한다.
v: 인수 포인터 (argv[0]..argv[n])가 포인터
의 배열로 전달된다. 인수의 수가 가변적
일때 사용된다.
e: 인수 env가 생성 프로세스에 전달되어 생
성 프로세스의 화경변수를 변경할 수 있게
한다. 접미사 e가 없으면 생성 프로세스는
모 프로세스의 환경을 물려받게 된다.

리 턴 값 에러가 없으면 spawn 함수는 생성 프로세스의
exit 상태를 리턴한다.
참 조 `abort`,`atexit`,`_exit`,`exec`,`exit`,
`_fpreset`,`searchpath`,`system`

쌇spawnlp
#함수 SPAWNLP

기 능 생성 (child) 프로세스를 만들고 실행시킨다.
구 문

```
#include <process.h>
#include <stdio.h>
int spawn(int mode, char *path, char *arg0,*arg1,...,
    argn,null);
int spawnle(int mode, char *path, char *arg0,
    *arg1,...,argn, null, char *envp[]);
int spawnlp(int mode, char *path, char *arg0,
    *arg1,...,argn,null);
int spawnlpe(int mode, char *path, char *arg0,
    *arg1,...,argn,null,char *envp[]);
int spawnv(int mode, char *path, char *argv[],
    char argp[]);
int spawnvp(int mode, char *path, char *argv[]);
```

```
int spawnvpe(int mode, char *path, char *argv[],
    char *argp[]);
```

프로토타입 `process.h`

기능 설명 spawn..에 속하는 함수들은 생성 프로세스(child process)로 알려진 다른 파일을 메모리에 올리고 실행시킨다.
이 때 생성 프로세스를 위한 충분한 메모리 공간이 있어야 한다.
mode 값은 실행시키는 모 프로세스가 생성 프로세스를 개시한 뒤에 어떤 작동을 하는지를 결정한다. mode 값은 아래와 같다.

P_WAIT 생성 프로세스가 실행을 끝낼 때까지 모프로세스의 실행을 중지한다.
P_NOWAIT 생성 프로세스를 샐행하면서 모 프로세스도 계속 실행한다.
P_OVERAY 생성 프로세스가 모 프로세스를 겹치도록 한다.`exec`을 호출하는 것과 같다.
주의 P_NOWAIT 는 아직 사용할 수 없다.

spawn.. 뒤에 접미사 l, v, p, e 등이 붙어서 각각의 함수명을 이루며 어떤 특정한 기능을 가지고 작동한다.

p: DOS PATH 환경변수가 지정하는 디렉토리에서 실행할 파일을 탐색한다.
l: 인수 포인터 (arg0, arg1, ...,argn)가 분리된 인수로서 전달되는데 보통 전달될 인수의 수를 미리 알고있을때 사용한다.
v: 인수 포인터 (argv[0]..argv[n])가 포인터의 배열로 전달된다. 인수의 수가 가변적일때 사용된다.
e: 인수 env가 생성 프로세스에 전달되어 생성 프로세스의 화경변수를 변경할 수 있게 한다. 접미사 e가 없으면 생성 프로세스는 모 프로세스의 환경을 물려받게 된다.

리 턴 값 에러가 없으면 spawn 함수는 생성 프로세스의 exit 상태를 리턴한다.

참　　조　`abort`,`atexit`,`_exit`,`exec`,`exit`,
　　`_fpreset`,`searchpath`,`system`

쌇spawnlpe
#함수 SPAWNLPE

기　　능　생성 (child) 프로세스를 만들고 실행시킨다.
구　　문

```
#include <process.h>
#include <stdio.h>
int spawn(int mode, char *path, char *arg0,*arg1,...,
    argn,null);
int spawnle(int mode, char *path, char *arg0,
    *arg1,...,argn, null, char *envp[]);
int spawnlp(int mode, char *path, char *arg0,
    *arg1,...,argn,null);
int spawnlpe(int mode, char *path, char *arg0,
    *arg1,...,argn,null,char *envp[]);
int spawnv(int mode, char *path, char *argv[],
    char argp[]);
int spawnvp(int mode, char *path, char *argv[]);
int spawnvpe(int mode, char *path, char *argv[],
    char *argp[]);
```

프로토타입　`process.h`
기능　설명　spawn..에 속하는 함수들은 생성 프로세스(ch
　　ild process)로 알려진 다른 파일을 메모리에
　　올리고 실행시킨다.
　　이 때 생성 프로세스를 위한 충분한 메모리
　　공간이 있어야 한다.
　　mode 값은 실행시키는 모 프로세스가 생성 프
　　로세스를 개시한 뒤에 어떤 작동을 하는지를
　　결정한다. mode 값은 아래와 같다.

　　P_WAIT　　생성 프로세스가 실행을 끝낼 때까
　　　　지 모프로세스의 실행을 중지한다.
　　P_NOWAIT　생성 프로세스를 샐행하면서 모 프
　　　　로세스도 계속 실행한다.
　　P_OVERAY　생성 프로세스가 모 프로세스를 겹
　　　　치도록 한다.`exec`을 호출하는 것

과 같다.
주의 P_NOWAIT 는 아직 사용할 수 없다.

spawn.. 뒤에 접미사 l, v, p, e 등이 붙어서
각각의 함수명을 이루며 어떤 특정한 기능을
가지고 작동한다.

p: DOS PATH 환경변수가 지정하는 디렉토리에
서 실행할 파일을 탐색한다.
l: 인수 포인터 (arg0, arg1, ...,argn)가 분
리된 인수로서 전달되는데 보통 전달될 인
수의 수를 미리 알고있을때 사용한다.
v: 인수 포인터 (argv[0]..argv[n])가 포인터
의 배열로 전달된다. 인수의 수가 가변적
일때 사용된다.
e: 인수 env가 생성 프로세스에 전달되어 생
성 프로세스의 화경변수를 변경할 수 있게
한다. 접미사 e가 없으면 생성 프로세스는
모 프로세스의 환경을 물려받게 된다.

리 턴 값 에러가 없으면 spawn 함수는 생성 프로세스의
exit 상태를 리턴한다.
참 조 `abort`,`atexit`,`_exit`,`exec`,`exit`,
`_fpreset`,`searchpath`,`system`

쌇spawnv
#함수 SPAWNV

기 능 생성 (child) 프로세스를 만들고 실행시킨다.
구 문

```
#include <process.h>
#include <stdio.h>
int spawn(int mode, char *path, char *arg0,*arg1,...,
    argn,null);
int spawnle(int mode, char *path, char *arg0,
    *arg1,...,argn, null, char *envp[]);
int spawnlp(int mode, char *path, char *arg0,
    *arg1,...,argn,null);
int spawnlpe(int mode, char *path, char *arg0,
    *arg1,...,argn,null,char *envp[]);
```

```
int spawnv(int mode, char *path, char *argv[],
    char argp[]);
int spawnvp(int mode, char *path, char *argv[]);
int spawnvpe(int mode, char *path, char *argv[],
    char *argp[]);
```

프로토타입 `process.h`

기능 설명 spawn..에 속하는 함수들은 생성 프로세스(child process)로 알려진 다른 파일을 메모리에 올리고 실행시킨다.
이 때 생성 프로세스를 위한 충분한 메모리 공간이 있어야 한다.
mode 값은 실행시키는 모 프로세스가 생성 프로세스를 개시한 뒤에 어떤 작동을 하는지를 결정한다. mode 값은 아래와 같다.

P_WAIT 생성 프로세스가 실행을 끝낼 때까지 모프로세스의 실행을 중지한다.
P_NOWAIT 생성 프로세스를 샐행하면서 모 프로세스도 계속 실행한다.
P_OVERAY 생성 프로세스가 모 프로세스를 겹치도록 한다.`exec`을 호출하는 것과 같다.
주의 P_NOWAIT 는 아직 사용할 수 없다.

spawn.. 뒤에 접미사 l, v, p, e 등이 붙어서 각각의 함수명을 이루며 어떤 특정한 기능을 가지고 작동한다.

p: DOS PATH 환경변수가 지정하는 디렉토리에서 실행할 파일을 탐색한다.
l: 인수 포인터 (arg0, arg1, ...,argn)가 분리된 인수로서 전달되는데 보통 전달될 인수의 수를 미리 알고있을때 사용한다.
v: 인수 포인터 (argv[0]..argv[n])가 포인터의 배열로 전달된다. 인수의 수가 가변적일때 사용된다.
e: 인수 env가 생성 프로세스에 전달되어 생성 프로세스의 화경변수를 변경할 수 있게 한다. 접미사 e가 없으면 생성 프로세스는 모 프로세스의 환경을 물려받게 된다.

리 턴 값 에러가 없으면 spawn 함수는 생성 프로세스의
exit 상태를 리턴한다.
참 조 `abort`,`atexit`,`_exit`,`exec`,`exit`,
`_fpreset`,`searchpath`,`system`

쌇spawnve
#함수 SPAWNVE

기 능 생성 (child) 프로세스를 만들고 실행시킨다.
구 문

```
#include <process.h>
#include <stdio.h>
int spawn(int mode, char *path, char *arg0,*arg1,...,
    argn,null);
int spawnle(int mode, char *path, char *arg0,
    *arg1,...,argn, null, char *envp[]);
int spawnlp(int mode, char *path, char *arg0,
    *arg1,...,argn,null);
int spawnlpe(int mode, char *path, char *arg0,
    *arg1,...,argn,null,char *envp[]);
int spawnv(int mode, char *path, char *argv[],
    char argp[]);
int spawnvp(int mode, char *path, char *argv[]);
int spawnvpe(int mode, char *path, char *argv[],
    char *argp[]);
```

프로토타입 `process.h`
기능 설명 spawn..에 속하는 함수들은 생성 프로세스(ch
ild process)로 알려진 다른 파일을 메모리에
올리고 실행시킨다.
이 때 생성 프로세스를 위한 충분한 메모리
공간이 있어야 한다.
mode 값은 실행시키는 모 프로세스가 생성 프
로세스를 개시한 뒤에 어떤 작동을 하는지를
결정한다. mode 값은 아래와 같다.

P_WAIT 생성 프로세스가 실행을 끝낼 때까
지 모프로세스의 실행을 중지한다.
P_NOWAIT 생성 프로세스를 샐행하면서 모 프

로세스도 계속 실행한다.
P_OVERAY 생성 프로세스가 모 프로세스를 겹치도록 한다.`exec`을 호출하는 것과 같다.
주의 P_NOWAIT 는 아직 사용할 수 없다.

spawn.. 뒤에 접미사 l, v, p, e 등이 붙어서 각각의 함수명을 이루며 어떤 특정한 기능을 가지고 작동한다.

p: DOS PATH 환경변수가 지정하는 디렉토리에서 실행할 파일을 탐색한다.
l: 인수 포인터 (arg0, arg1, ...,argn)가 분리된 인수로서 전달되는데 보통 전달될 인수의 수를 미리 알고있을때 사용한다.
v: 인수 포인터 (argv[0]..argv[n])가 포인터의 배열로 전달된다. 인수의 수가 가변적일때 사용된다.
e: 인수 env가 생성 프로세스에 전달되어 생성 프로세스의 화경변수를 변경할 수 있게 한다. 접미사 e가 없으면 생성 프로세스는 모 프로세스의 환경을 물려받게 된다.

리 턴 값 에러가 없으면 spawn 함수는 생성 프로세스의 exit 상태를 리턴한다.
참 조 `abort`,`atexit`,`_exit`,`exec`,`exit`, `_fpreset`,`searchpath`,`system`

쌇spawnvp
#함수 SPAWNVP

기 능 생성 (child) 프로세스를 만들고 실행시킨다.
구 문

```
#include <process.h>
#include <stdio.h>
int spawn(int mode, char *path, char *arg0,*arg1,...,
    argn,null);
int spawnle(int mode, char *path, char *arg0,
    *arg1,...,argn, null, char *envp[]);
int spawnlp(int mode, char *path, char *arg0,
```

*arg1,...,argn,null);
int spawnlpe(int mode, char *path, char *arg0,
*arg1,...,argn,null,char *envp[]);
int spawnv(int mode, char *path, char *argv[],
char argp[]);
int spawnvp(int mode, char *path, char *argv[]);
int spawnvpe(int mode, char *path, char *argv[],
char *argp[]);

프로토타입 `process.h`
기능 설명 spawn..에 속하는 함수들은 생성 프로세스(ch
ild process)로 알려진 다른 파일을 메모리에
올리고 실행시킨다.
이 때 생성 프로세스를 위한 충분한 메모리
공간이 있어야 한다.
mode 값은 실행시키는 모 프로세스가 생성 프
로세스를 개시한 뒤에 어떤 작동을 하는지를
결정한다. mode 값은 아래와 같다.

P_WAIT 생성 프로세스가 실행을 끝낼 때까
지 모프로세스의 실행을 중지한다.
P_NOWAIT 생성 프로세스를 샐행하면서 모 프
로세스도 계속 실행한다.
P_OVERAY 생성 프로세스가 모 프로세스를 겹
치도록 한다.`exec`을 호출하는 것
과 같다.
주의 P_NOWAIT 는 아직 사용할 수 없다.

spawn.. 뒤에 접미사 l, v, p, e 등이 붙어서
각각의 함수명을 이루며 어떤 특정한 기능을
가지고 작동한다.

p: DOS PATH 환경변수가 지정하는 디렉토리에
서 실행할 파일을 탐색한다.
l: 인수 포인터 (arg0, arg1, ...,argn)가 분
리된 인수로서 전달되는데 보통 전달될 인
수의 수를 미리 알고있을때 사용한다.
v: 인수 포인터 (argv[0]..argv[n])가 포인터
의 배열로 전달된다. 인수의 수가 가변적
일때 사용된다.
e: 인수 env가 생성 프로세스에 전달되어 생

성 프로세스의 화경변수를 변경할 수 있게 한다. 접미사 e가 없으면 생성 프로세스는 모 프로세스의 환경을 물려받게 된다.

리 턴 값 에러가 없으면 spawn 함수는 생성 프로세스의 exit 상태를 리턴한다.
참 조 `abort`,`atexit`,`_exit`,`exec`,`exit`, `_fpreset`,`searchpath`,`system`

쌿spawnvpe
#함수 SPAWNVPE

기 능 생성 (child) 프로세스를 만들고 실행시킨다.
구 문

```
#include <process.h>
#include <stdio.h>
int spawn(int mode, char *path, char *arg0,*arg1,...,
    argn,null);
int spawnle(int mode, char *path, char *arg0,
    *arg1,...,argn, null, char *envp[]);
int spawnlp(int mode, char *path, char *arg0,
    *arg1,...,argn,null);
int spawnlpe(int mode, char *path, char *arg0,
    *arg1,...,argn,null,char *envp[]);
int spawnv(int mode, char *path, char *argv[],
    char argp[]);
int spawnvp(int mode, char *path, char *argv[]);
int spawnvpe(int mode, char *path, char *argv[],
    char *argp[]);
```

프로토타입 `process.h`
기능 설명 spawn..에 속하는 함수들은 생성 프로세스(child process)로 알려진 다른 파일을 메모리에 올리고 실행시킨다.
이 때 생성 프로세스를 위한 충분한 메모리 공간이 있어야 한다.
mode 값은 실행시키는 모 프로세스가 생성 프로세스를 개시한 뒤에 어떤 작동을 하는지를 결정한다. mode 값은 아래와 같다.

P_WAIT 생성 프로세스가 실행을 끝낼 때까지 모프로세스의 실행을 중지한다.
P_NOWAIT 생성 프로세스를 샐행하면서 모 프로세스도 계속 실행한다.
P_OVERAY 생성 프로세스가 모 프로세스를 겹치도록 한다.`exec`을 호출하는 것과 같다.
주의 P_NOWAIT 는 아직 사용할 수 없다.

spawn.. 뒤에 접미사 l, v, p, e 등이 붙어서 각각의 함수명을 이루며 어떤 특정한 기능을 가지고 작동한다.

p: DOS PATH 환경변수가 지정하는 디렉토리에서 실행할 파일을 탐색한다.
l: 인수 포인터 (arg0, arg1, ...,argn)가 분리된 인수로서 전달되는데 보통 전달될 인수의 수를 미리 알고있을때 사용한다.
v: 인수 포인터 (argv[0]..argv[n])가 포인터의 배열로 전달된다. 인수의 수가 가변적일때 사용된다.
e: 인수 env가 생성 프로세스에 전달되어 생성 프로세스의 화경변수를 변경할 수 있게 한다. 접미사 e가 없으면 생성 프로세스는 모 프로세스의 환경을 물려받게 된다.

리 턴 값 에러가 없으면 spawn 함수는 생성 프로세스의 exit 상태를 리턴한다.
참 조 `abort`,`atexit`,`_exit`,`exec`,`exit`, `_fpreset`,`searchpath`,`system`

쌇system
#함수 SYSTEM

기 능 DOS 명령을 실행한다.
구 문 int system(const char *command);
프로토타입 `stdlib.h`,`process.h`
기능 설명 command로 전달된 DOS 명령, 배치 파일, 혹은 다른 프로그램을 C 프로그램 내에서 실행 시키기 위해서 COMMAND.COM 파일을 호출한다.

실행하기 위한 프로그램은 현재 디렉토리에
있거나 DOS의 환경 PATH 문자열에 표시된 디
렉토리에 있어야 한다.

COMMAND.COM 파일을 찾기위해 COMSPEC 변수가
사용되므로 COMMAND.COM 파일이 현재 디렉토
리에 있을 필요는 없다.

리 턴 값 에러가 없으면 0을, 에러가 발생하면 -1을 리
턴한다.

이 식 성 UNIX 시스템상에서 사용할 수 있으며 ANSI-C
와도 호환된다.

참 조 `exec`,`_fpreset`,`searchpath`,`spawn`

쌿setjmp.h
#헤더파일 SETJMP.H

함수

`longjmp` `setjmp`

쌿longjmp
#함수 LONGJMP

기 능 local을 벗어나는 분기를 한다.

구 문 #include <setjmp.h>
void longjmp(jmp_buf jmpb, int retval);

프로토타입 `setjmp.h`

기능 설명 longjmp 호출은 인수 jmpb로 마지막 호출한
setjmp가 가지고 있는 작업 상태를 복구한다.
이 때 이 함수는 setjmp가 retval 값으로 리
턴한 것과 같이 리턴한다.
작업상태는 다음과 같다.

* 모든 세그먼트 레지스터(CS, DS, ES, SS)
* 레지스터 변수(SI, DI)
* 스택 포인터(SP)
* 베이스 포인터(BP)
* 플래그

작업 상태는 setjmp와 longjmp가 동일 루틴을
수행하는 데 사용될 수 있을 정도로 완벽하다

이들 루틴은 프로그램이 저차원부 루틴에서 발생하는 예외 사항과 에러를 처리하는데 유용하다.
setjmp는 longjmp이전에 수행되어야 한다. setjmp를 호출해서 jmpb를 설정한 루틴은 계속 작동해야 하며 longjmp가 호출되기 전에 리턴할 수가 없다.

리 턴 값 없음
이 식 성 UNIX 시스템상에서 사용할 수 있으며 ANSI-C와도 호환된다.
참 조 `setjmp`,`signal`

쌇setjmp
#함수 SETJMP

기 능 nonlocal goto를 수행한다.
구 문 #include <setjmp.h>
int setjmp(jmp_buf jmpb);
프로토타입 `setjmp.h`
기능 설명 jmpb에서 task state를 가지며 0을 리턴한다.
이후에 jmpb를 가지고 longjmp를 호출하면 갖고 있는 task state가 복구되고 setjmp가 val 값을 가지고 리턴했던 것과 같이 리턴된다.
수행 상태는 다음과 같다.

* 모든 세그먼트 레지스터(CS, DS, ES, SS)
* 레지스터 변수(SI, DI)
* 스택 포인터(SP)
* 베이스 포인터(BP)
* 플래그

리 턴 값 처음 호출되었을 때 0을 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있으며, ANSI-C와도 호환된다.
참 조 `longjmp`,`signal`

쌇share.h
#헤더파일 SHARE.H

SHARE.H에는 파일공유에 대한 상수가 정의되어 있다.

SH_COMPAT SH_DENYRD
SH_DENYNO SH_DENYRW
SH_DENYNONE SH_DENYWR

쌇signal.h
#헤더파일 SIGNAL.H

함수

`raise` `signal`
쌇raise
#함수 RAISE

기 능 실행중인 프로그램에 소프프트웨어의 신호를 전송한다.
구 문 #include <signal.h>
int raise(int sig);
프로토타입 `signal.h`
기능 설명 sig 신호를 프로그램에 전송한다. 만일 프로그램이 sig로 지정된 신호 조정기를 설치 하였다면 그 조정기가 실행된다. 만일 아무런 조정기도 설치되지 않았다면 시그널형의 디폴트 처리가 수행된다. SIGNAL.H에 정의된 시스널 형은 다음과 같다.

신호	의미
SIGABRT	이상 종료(*)
SIGFPF	부동 소숫점 연산 에러
SIGILL	부적당한 명령(#)
SIGINT	콘트롤 브레이크 인터럽트
SIGSEGV	주기억에의 부당한 억세스(#)
SIGTERM	프로그램 종료 요구(*)

'*'로 표기된 신호 형태의 작동이 정상적인 경우 DOS나 TURBO-C에 의해 생성되지 않는다. 그러나 raise로 생성될 수 있다.
'#'로 표기된 신호 형태는 8088또는 8086프로세서에서 비동기적으로 생성될 수 없지만 그 외 다른 프로세서에서는 생성 가능하다.
리 턴 값 에러가 없으면 0을 리턴하고 그렇지 않으면 0

이외의 값을 리턴한다.
이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C와 도 호환된다.
참 조 `abort`,`signal`

쌇signal
#함수 SIGNAL

기 능 신호 조정 작동을 지정한다.
구 문 #include <signal.h>
void (*signal(int sig, void (*func)
(int sig[,int subcode])))(int);
프로토타입 `signal.h`
기능 설명 신호번호 sig가 접수 되었을 때 처리 방법을 결정한다. 사용자는 사용자 지정 처리 루틴을 설치하거나 사전 정의된 루틴을 이용할 수 가 있다.

신호	의미
SIGABRT	이상 종료(*)
SIGFPF	부동 소숫점 연산 에러
SIGILL	부적당한 명령(#)
SIGINT	콘트롤 브레이크 인터럽트
SIGSEGV	주기억에의 부당한 억세스(#)
SIGTERM	프로그램 종료 요구(*)

리 턴 값 호출에 에러가 없으면 signal은 지정된 신호 형태에 대한 이전의 처리루틴의 주소를 리턴 한다. 에러시에는 SIG_ERR이 리턴되며, 외부 변수`errno`는 EINVAL로 설정된다.
이 식 성 ANSI-C와 호환된다.
참 조 `abort`,`_control87`,`ctrlbrk`,`exit`, `longjmp`,`raise`,`setjmp`

쌇stdarg.h
#헤더파일 STDARG.H

함수

`va_arg` `va_end` `va_start`

상수, 데이타형, 파리미터형

`va_list`

쌇va_arg
#함수 VA_ARG

기　　능　가변 인수 리스트를 처리한다.
구　　문　#include <stdarg.h>
　　void va_start(va_list param, lastfix);
　　void va_arg(va_list param, type);
　　void va_end(va_list param);
프로토타입　`stdarg.h`
기능 설명　C의 어떤 함수들은`vfprint`나`vprintf`처럼 파라미터 이외에도 가변 인수 리스트를 가질 수 있다. va_... 매크로는 이러한 가변 인수를 간단히 다루는 방법이다.
　　이 함수들은 인수를 받는 함수가 실제 전달된 인수의 수를 모를 경우에 하나씩 처리하게 해 준다.

* va_list
　　이 배열은 va_arg와 va_end가 필요로 하는 정보를 담고 있다. 가변 인수를 가지는 호출 함수는 va_list형의 인수 param을 선언해야 한다.
* va_start
　　매크로로 구현된 이루틴은 param이 그 함수에 전달된 첫번째 가변 인수를 가리키도록 한다.
　　va_start는 반드시 va_arg나 va_end보다 먼저 호출되어야 한다.
　　va_start는 두 개의 파라미터(param과 lastfix)를 가진다.
* va_arg
　　매크로로 수행되는 이루틴은 전달되는 인수와 같은 형과 값을 갖는 식으로 전개된다. va_arg에 주어지는 변수 param은 va_start를 초기화하는 param과 동일하여야 한다.
* va_end
　　이 매크로는 호출된 함수가 정상적으로 리턴

하도록 도와준다. va_end는 va_arg가 모든 인수를 읽은 후에 호출되어야 한다. 이렇게 하지 않을경우 프로그램은 정의되지 않은 이상한 작동을 하게 된다.

리 턴 값 va_start와 va_end는 리턴값이 없다. va_arg는 리스트에 있는 현재 인수를 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있다.
참 조 `vfprintf`,`vprintf`,`vscanf`

쌇va_end
#함수 VA_END

기 능 가변 인수 리스트를 처리한다.
구 문 #include <stdarg.h>
void va_start(va_list param, lastfix);
void va_arg(va_list param, type);
void va_end(va_list param);
프로토타입 `stdarg.h`
기능 설명 C의 어떤 함수들은`vfprint`나`vprintf`처럼 파라미터 이외에도 가변 인수 리스트를 가질 수 있다. va_... 매크로는 이러한 가변 인수를 간단히 다루는 방법이다.
이 함수들은 인수를 받는 함수가 실제 전달된 인수의 수를 모를 경우에 하나씩 처리하게 해준다.

* va_list
이 배열은 va_arg와 va_end가 필요로 하는 정보를 담고 있다. 가변 인수를 가지는 호출 함수는 va_list형의 인수 param을 선언해야 한다.
* va_start
매크로로 구현된 이루틴은 param이 그 함수에 전달된 첫번째 가변 인수를 가리키도록 한다. va_start는 반드시 va_arg나 va_end보다 먼저 호출되어야 한다.
va_start는 두 개의 파라미터(param과 lastfix)를 가진다.
* va_arg
매크로로 수행되는 이루틴은 전달되는 인수와

같은 형과 값을 갖는 식으로 전개된다. va_arg에 주어지는 변수 param은 va_start를 초기화하는 param과 동일하여야 한다.

* va_end

이 매크로는 호출된 함수가 정상적으로 리턴하도록 도와준다. va_end는 va_arg가 모든 인수를 읽은 후에 호출되어야 한다. 이렇게 하지 않을경우 프로그램은 정의되지 않은 이상한 작동을 하게 된다.

리 턴 값 va_start와 va_end는 리턴값이 없다. va_arg는 리스트에 있는 현재 인수를 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있다.
참 조 `vfprintf`,`vprintf`,`vscanf`

쌿va_start
#함수 VA_START

기 능 가변 인수 리스트를 처리한다.
구 문 #include <stdarg.h>
void va_start(va_list param, lastfix);
void va_arg(va_list param, type);
void va_end(va_list param);
프로토타입 `stdarg.h`
기능 설명 C의 어떤 함수들은`vfprint`나`vprintf`처럼 파라미터 이외에도 가변 인수 리스트를 가질 수 있다. va_... 매크로는 이러한 가변 인수를 간단히 다루는 방법이다.
이 함수들은 인수를 받는 함수가 실제 전달된 인수의 수를 모를 경우에 하나씩 처리하게 해준다.

* va_list

이 배열은 va_arg와 va_end가 필요로 하는 정보를 담고 있다. 가변 인수를 가지는 호출 함수는 va_list형의 인수 param을 선언해야 한다.

* va_start

매크로로 구현된 이루틴은 param이 그 함수에 전달된 첫번째 가변 인수를 가리키도록 한다. va_start는 반드시 va_arg나 va_end보다 먼저

호출되어야 한다.
va_start는 두 개의 파라미터(param과 lastfi
x)를 가진다.

* va_arg
매크로로 수행되는 이루틴은 전달되는 인수와 같은 형과 값을 갖는 식으로 전개된다. va_ar
g에 주어지는 변수 param은 va_start를 초기화하는 param과 동일하여야 한다.

* va_end
이 매크로는 호출된 함수가 정상적으로 리턴하도록 도와준다. va_end는 va_arg가 모든 인수를 읽은 후에 호출되어야 한다. 이렇게 하지 않을경우 프로그램은 정의되지 않은 이상한 작동을 하게 된다.

리 턴 값 va_start와 va_end는 리턴값이 없다. va_arg는 리스트에 있는 현재 인수를 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있다.
참 조 `vfprintf`,`vprintf`,`vscanf`

쌇va_list
#데이타형 VA_LIST

기 능 가변 인수 리스트를 처리한다.
구 문 #include <stdarg.h>
void va_start(va_list param, lastfix);
void va_arg(va_list param, type);
void va_end(va_list param);
프로토타입 `stdarg.h`
기능 설명 C의 어떤 함수들은`vfprint`나`vprintf`처럼 파라미터 이외에도 가변 인수 리스트를 가질 수 있다. va_... 매크로는 이러한 가변 인수를 간단히 다루는 방법이다.
이 함수들은 인수를 받는 함수가 실제 전달된 인수의 수를 모를 경우에 하나씩 처리하게 해준다.

* va_list
이 배열은 va_arg와 va_end가 필요로 하는 정보를 담고 있다. 가변 인수를 가지는 호출 함수는 va_list형의 인수 param을 선언해야 한

다.

* va_start
  매크로로 구현된 이루틴은 param이 그 함수에 전달된 첫번째 가변 인수를 가리키도록 한다. va_start는 반드시 va_arg나 va_end보다 먼저 호출되어야 한다.
  va_start는 두 개의 파라미터(param과 lastfix)를 가진다.
* va_arg
  매크로로 수행되는 이루틴은 전달되는 인수와 같은 형과 값을 갖는 식으로 전개된다. va_arg에 주어지는 변수 param은 va_start를 초기화하는 param과 동일하여야 한다.
* va_end
  이 매크로는 호출된 함수가 정상적으로 리턴하도록 도와준다. va_end는 va_arg가 모든 인수를 읽은 후에 호출되어야 한다. 이렇게 하지 않을경우 프로그램은 정의되지 않은 이상한 작동을 하게 된다.

리 턴 값 va_start와 va_end는 리턴값이 없다. va_arg는 리스트에 있는 현재 인수를 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있다.
참 조 `vfprintf`,`vprintf`,`vscanf`

쌇stddef.h
#헤더파일 STDDEF

상수, 데이타형, 파리미터형

`errno` `ptrdiff_t` `NUll`
`size_t`

쌇stdio.h
#헤더파일 STDIO.H

`clearerr` `fclose` `fcloseall`
`fdopen` `feof` `ferror`
`fflush` `fgetc` `fgetchar`
`fgetpos` `fgets` `fileno`
`flushall` `fopen` `fprintf`

`fputc` `fputchar` `fputs`
`fread` `freopen` `fscanf`
`fseek` `fsetpos` `ftell`
`fwrite` `getc` `getchar`
`gets` `getw` `perror`
`printf` `putc` `putchar`
`puts` `putw` `remove`
`rename` `rewind` `scanf`
`setbuf` `setvbuf` `sprintf`
`sscanf` `_strerror` `strerror`
`tmpfile` `tmpnam` `ungetc`
`unlink` `vfprintf` `vfscanf`
`vprintf` `vscanf` `vsprintf`
`vsscanf`

쌿clearerr
#함수 CLEARERR

기　　능　파일의 에러 상태를 다시 설정한다.
구　　문　#include <stdio.h>
　　void clearerr(FILE *stream);
프로토타입　`stdio.h`
기능 설명　파일 스트림에 관한 에러 및 EOF 식별자를 0으로 재설정한다. 한번 에러 식별자가 설정되면 clearerr나 rewind가 호출될 때까지 스트림 작동은 계속해서 에러를 리턴하게 된다 EOF 식별자는 입력 작동과 동시에 재설정 된다. 여기서 stream으로 지정된 파일이 다른 함수 실행시에 에러로 된 경우에 clearerr() 은 에러를 해제시킨다.
리 턴 값　없음
이 식 성　UNIX 시스템에상에서 사용할 수 있으며 ANSI-C와 호환된다.
참　　조　`eof`,`feof`,`ferror`,`perror`,`rewind`

쌿fclose
#함수 FCLOSE

기　　능　스트림을 닫는다.
구　　문　#include <stdio.h>
　　int fclose(FILE *stream);

프로토타입 `stdio.h`
기능 설명 지명된 스트림을 닫는다. 일반적으로 스트림
과 관련된 모든 버퍼는 닫기 전에 초기상태로
된다. 시스템에 할당된 버퍼는 파일 닫음에서
해제된다. 그러나 setbuf혹은 setvbuf로 지정
된 버퍼는 자동으로 해제되지는 않는다.
리 턴 값 에러가 없으면 0을 리턴한다. 에러가 발생한
경우에는 EOF가 리턴된다.
이 식 성 UNIX 시스템에상에서 사용할 수 있으며 ANSI-
C와 호환된다.
참 조 `close`,`fcloseall`,`fdopen`,`fflush`,
`flushall`,`fopen`,`freopen`

쌇fcloseall
#함수 FCLOSEALL

기 능 개방된 스트림을 닫는다.
구 문 int fcloseall(void);
프로토타입 `stdio.h`
기능 설명 stdin,stdout,stdprn,stderr,stdaux를 제외한
개방되어 있는 모든 스트림을 닫는다.
리 턴 값 닫혀진 스트림의 총수를 리턴하며 에러 발생
시 EOF를 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있다.
참 조 `fclose`,`fdopen`,`flushall`,`fopen`,
`freopen`

쌇fdopen
#함수 FDOPEN

기 능 스트림과 파일 handle을 연결한다.
구 문 #include <stdio.h>
FILE *fdopen(int handle, char *type);
프로토타입 `stdio.h`
기능 설명 스트림을 creat, dup, dup2, open으로부터 얻
은 파일 handle과 연결한다. 스트림의 형은
개방형 handle의 모드와 일치해야 한다. fdop
en 호출에서 사용되는 type문자열은 다음과
같은 값을 갖는다.
---
r 읽기방식(read mode), 읽어들일 파일이

이미 존재해야 함
w 쓰기방식(write mode), 파일이 없으면 주어진 이름으로 새로 만들어 지고 있으면 초기화 됨
a 추가방식(append mode), 파일이 없으면 새로 만들어 지고, 있으면 현재 위치가 파일의 끝이 됨
r+ 파일 갱신을 위하여 오픈(읽기/쓰기)
w+ 파일 갱신을 위하여 새로만듬
a+ 파일 갱신을 위하여 오픈(읽기/쓰기), 파일이 없으면 새로 만들어지고, 있으면 현재 위치를 파일의 끝으로 한다.

리 턴 값 에러없이 수행되면 fdopen은 새로이 개방된 스트림에 대한 포인터를 리턴한다. 에러가 발생하면 null을 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있다.
참 조 `fclose`,`fopen`,`freopen`,`open`

쌇feof
#함수 FEOF

기 능 스트림상에서 파일의 종료(EOF)를 탐지한다.
구 문 #include <stdio.h>
inr eof(FILE *stream);
프로토타입 `stdio.h`
기능 설명 EOF 지시자를 위해 주어진 스트림을 조사하는 매크로이다. 일단 지시자가 설정되면 rewind 가 호출되거나 파일이 닫힐 때까지 읽기 작동은 이 지시자를 리턴하게 된다. EOF 지시자는 각 입력 작동과 함께 rewind된다.
리 턴 값 EOF 지시자가 지정된 스트림상의 마지막 입력 직동에서 탐지되면 0이외의 값을 리턴하고, EOF를 만나지 못하면 0이 리턴된다.
이 식 성 UNIX 시스템에상에서 사용할 수 있으며 ANSI-C와 호환된다.
참 조 `clearerr`,`eof`,`ferror`,`perror`

쌇ferror
#함수 FERROR

기 능 스트림상에서 에러를 탐지한다.
구 문 #include <stdio.h>
int ferror(FILE *stream);
프로토타입 `stdio.h`
기능 설명 읽기나 쓰기의 에러상태를 알기 위하여 주어진 스트림의 에러상태를 조사하는 매크로 이다. 스트림의 에러지시자가 설정되면 clearerr이나 rewind가 호출될 때까지 아니면 이 스트림이 닫힐 때까지 해제되지 않고 남아있게 된다.
리 턴 값 지정된 스트림에 에러가 탐지되면 0이외의 값이 리턴된다.
이 식 성 UNIX 시스템에상에서 사용할 수 있으며 ANSI-C와 호환된다.
참 조 `clearerr`,`eof`,`feof`,`fopen`,`gets`, `perror`

쌇fflush
#함수 FFLUSH

기 능 스트림 버퍼를 클리어한다.
구 문 #include <stdio>
int fflush(FILE *stream);
프로토타입 `stdio.h`
기능 설명 주어진 스트림이 출력용으로 개방되어 있으면 fflush는 파일 stream의 출력 버퍼 내용을 연관된 파일로 내보낸다. 만일 stream이 입력용으로 개방되어 있다면 fflush는 그 버퍼의 내용을 클리어한다. 스트림은 fflush가 호출된 후에도 개방상태로 남는다. fflush는 버퍼가 없는 스트림에 대해서는 영향을 미치지 않는다.
리 턴 값 에러 없이 수행되면 0을 리턴하고, 에러 발생시 EOF를 리턴한다.
이 식 성 UNIX 시스템에상에서 사용할 수 있으며 ANSI-C와 호환된다.
참 조 `fclose`,`flushall`,`setbuf`,`setvbuf`

쌇fgetc
#함수 FGETC

기　　능　스트림으로부터 문자를 읽어들인다.
구　　문　#include <stdio.h>
　　int fgetc(FILE *stream);
프로토타입　`stdio.h`
기능 설명　지정된 입력 스트림에서 다음 문자를 리턴한다.
리 턴 값　에러없이 이루어지면 fgetc는 문자를 정수값으로 변환한 후에 읽어들인 문자를 리턴한다. EOF를 읽어들인 경우나 에러가 발생한 경우에는 EOF를 리턴한다.
이 식 성　UNIX 시스템에상에서 사용할 수 있으며 ANSI-C와 호환된다.
참　　조　`fgetchar`,`fputc`,`getc`,`getch`, `getchar`,`getche`,`ungetc`,`ungetch`

쌇fgetchar
#함수 FGETCHAR

기　　능　stdin으로부터 문자를 읽어들인다.
구　　문　int fgechar(void);
프로토타입　`stdio.h`
기능 설명　stdin으로부터 다음 문자를 리턴한다. 이것은 fgetc(stdin)으로 정의된 것과 똑같이 작동한다.
리 턴 값　에러없이 이루어지면 문자를 정수값으로 변환한 후에 읽어들인 문자를 리턴한다. EOF를 읽어들인 경우나 에러가 발생한 경우에는 EOF를 리턴한다.
이 식 성　UNIX 시스템상에서 사용할 수 있다.
참　　조　`fgetc`,`fputchar`,`getchar`

쌇fgetpos
#함수 FGETPOS

기　　능　현재의 파일 포인터를 저장한다.
구　　문　#include <stdio.h>
　　int fgetpos(FILE *stream,fops_t *pos);
프로토타입　`stdio.h`
기능 설명　주어진 스트림에 연관되어 있는 현재 파일 포인터를 pos에서 지정하고 있는 곳에 저장하는 역할을 한다. fpos_t는 long형으로 정의되어

있다.
리 턴 값 에러없이 수행되면 0을 리턴한다, 에러가 발생하면 0이외의 값을 리턴하고 errno를 EBADF나 EINVAL로 설정한다.
이 식 성 ANSI-C와 호환성이 있다.
참 조 `fseek`,`fsetpos`,`ftell`,`tell`

쌇fgets
#함수 FGETS

기 능 스트림으로부터 문자열을 읽어들인다.
구 문 #include <stdio.h>
char *fgets(char *s,int n,FILE *stream);
프로토타입 `stdio.h`
기능 설명 stream으로부터 문자를 읽어들여 문자열 s에 저장한다. 새로운 행의 문자를 읽어들일때, n-1 문자를 읽어들인 때에는 읽기를 중지한다 또한 새로운 행의 문자를 문자열 안에 위치시키지 않는다. s로 읽어들인 마지막 문자에는 null 종료문자가 위치하게 된다.
리 턴 값 에러없이 수행되면 s가 가리키는 문자열을 리턴하고 EOF나 에러에는 EOF를 리턴한다.
이 식 성 UNIX 시스템에상에서 사용할 수 있으며 ANSI-C와 호환된다.
참 조 `cgets`,`fputs`,`gets`

쌇fileno
#함수 FILENO

기 능 파일 handle을 갖는다.
구 문 #include <stdio.h>
int fileno(FILE *stream);
프로토타입 `stdio.h`
기능 설명 주어진 스트림에 대한 파일 handle을 리턴하는 매크로이다. stream이 한 개 이상의 handle을 갖는다면 fileno는 최초로 개방될 때 스트림에 지정된 handle을 리턴한다.
리 턴 값 스트림과 연관된 정수 파일 handle을 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있다.
참 조 `fdopen`,`fopen`,`freopen`

쌿flushall
#함수 FLUSHALL

기　　능　모든 스트림의 버퍼을 클리어한다.
구　　문　int flushall(void);
프로토타입　`stdio.h`
기능 설명　flushall은 개방되어 있는 입력스트림과 연관
된 모든 버퍼를 클리어 시키고 개방되어 있는
모든 출력 스트림 버퍼의 내용을 대응되는 파
일에 작성한다.
flushall 다음에 오는 읽기 연산은 입력 파일
로부터 새로운 데이타를 버퍼로 읽어온다. fl
ushall이 호출된 후에도 스트림은 개방 상태
로 있게된다.
리 턴 값　flushall은 개방 입/출력 스트림의 수를 정수
로 리턴한다.
이 식 성　UNIX 시스템상에서 사용할 수 있다.
참　　조　`fclose`,`fcloseall`,`fflush`

쌿fopen
#함수 FOPEN

기　　능　스트림을 개방한다.(파일오픈)
구　　문　#include <stdio.h>
FILE *fopen(const char *filename,
const char *mode);
프로토타입　`stdio.h`
기능 설명　filename을 개방하고 이를 스트림과 연관시키
며 스트림을 확인하는데 사용되는 포인터를
리턴한다.
fopen을 호출하는데 사용되는 mode 문자열은
다음 값중의 하나가 된다.
---
r　읽기방식(read mode), 읽어들일 파일이
이미 존재해야 함
w　쓰기방식(write mode), 파일이 없으면 주
어진 이름으로 새로 만들어 지고 있으면
초기화 됨
a　추가방식(append mode), 파일이 없으면
새로 만들어 지고, 있으면 현재 위치가

파일의 끝이 됨
r+ 파일 갱신을 위하여 오픈(읽기/쓰기)
w+ 파일 갱신을 위하여 새로만듬
a+ 파일 갱신을 위하여 오픈(읽기/쓰기), 파일이 없으면 새로 만들어지고, 있으면 현재 위치를 파일의 끝으로 한다.

주어진 파일이 텍스트 모드로 개방되거나 생성되기 위해서는 mode 문자열에 t를 추가하고 (rt,w+t 등), 이진모드로 지정하기 위해서는 b를 추가한다.

리 턴 값 에러없이 수행되면 새로 개방된 스트림에 대한 포인터를 리턴한다. 에러발생시 null을 리턴한다.

이 식 성 UNIX 시스템에상에서 사용할 수 있으며 ANSI-C와 호환된다.

참 조 `creat`,`dup`,`fclose`,`fdopen`,`ferror`, `fopen`,`fread`,`freopen`,`fseek`,`fwrite`, `open`,`fwrite`,`open`,`rewind`,`setbuf`, `setmode`

쌿fprintf

#함수 FPRINTF

기 능 포맷된 출력을 스트림에 보낸다.

구 문 #include <stdio.h>
int fprintf(FILE *stream,const char *format[,argument, ...]);

프로토타입 `stdio.h`

기능 설명 일련의 인자를 받아서 format이 가리키는 포맷 문자열에 있는 포맷 지정에 적용하고 포맷된 데이타를 스트림에 출력한다. 이 때 인자와 같은 수의 형식이 있어야 한다. 포맷 지정에 대한 사항은`printf`를 참고한다.

리 턴 값 출력된 바이트 수를 리턴한다. 에러 발생시 EOF를 리턴한다.

이 식 성 UNIX 시스템에상에서 사용할 수 있으며 ANSI-C와 호환된다.

참 조 `cprintf`,`fscanf`,`printf`,`putc`,`sprintf`

쌿fputc

#함수 FPUTC

기　　능　스트림에 한 개의 문자를 써 넣는다.
구　　문　#include <stdio.h>
　　int fputc(int c,FILE *stream);
프로토타입　`stdio.h`
기능 설명　문자 c를 지정된 스트림에 출력한다.
리 턴 값　출력이 수행되면 문자 c를 리턴한다. 에러 발생시 EOF를 리턴한다.
이 식 성　UNIX 시스템에상에서 사용할 수 있으며 ANSI-C와 호환된다.
참　　조　`fgetc`,`putc`

쌇fputchar
#함수 FPUTCHAR

기　　능　stdout에 한 개의 문자를 출력한다.
구　　문　#include <stdio.h>
　　int fputchar(int c);
프로토타입　`stdio.h`
기능 설명　문자 c를 stdout에 출력한다. fpuchar(c)는 fputc(c,stdout)와 같다.
리 턴 값　에러없이 수행되면 문자 c를 리턴하고 에러 발생시 EOF를 리턴한다.
이 식 성　UNIX 시스템상에서 사용할 수 있다.
참　　조　`fderchar`,`putchar`

쌇fputs
#함수 FPUTS

기　　능　스트림에 한 개의 문자열을 출력한다.
구　　문　#include <stdio.h>
　　int fputs(const char *s,FILE *stream);
프로토타입　`stdio.h`
기능 설명　null 종료문자열 s를 주어진 스트림에 복사해 온다. 여기서는 새로운 행의 문자열을 추가하지 않으며 null 문자 자신도 복사되지 않는다
리 턴 값　에러없이 수행되면 쓰여진 마지막 문자가 리턴되고 에러발생시 EOF가 리턴된다.
이 식 성　UNIX 시스템에상에서 사용할 수 있으며 ANSI-C와 호환된다.

참　　조　`fgets`,`gets`,`puts`

쌇fread
#함수 FREAD

기　　능　스트림으로부터 데이터를 읽는다.
구　　문　#include <stdio.h>
　　size_t fread(*ptr, size_t size,size_t n,
　　FILE *stream);
프로토타입　`stdio.h`
기능 설명　주어진 입력 스트림으로부터 각각 size 바이
　　트의 길이로 데이타를 n개 읽어들인 후 그 데
　　이타를 포인터 ptr에서 지정하는 블록에 가져
　　온다.
　　읽어들인 총 바이트수는 n * size가 된다.
리 턴 값　에러없이 수행되면 실제로 읽혀진 항목수를
　　리턴한다. 에러나 EOF시 0이 리턴된다.
이 식 성　UNIX 시스템에상에서 사용할 수 있으며 ANSI-
　　C와 호환된다.
참　　조　`fopen`,`fwrite`,`printf`,`read`

쌇freopen
#함수 FREOPEN

기　　능　스트림을 대체한다.
구　　문　#include <stdio.h>
　　FILE *freopen(const char *filename,
　　const char *mode,
　　FILE *stream);
프로토타입　`stdio.h`
기능 설명　개방 stream 대신 지정된 파일로 대체한다.
　　따라서 이것은 사전 개방되어 있는 스트림을
　　filename에서 지정한 파일로 치환하는 함수이
　　다. 개방 처리에 무관하게 원래의 stream은
　　닫혀지게 된다.

　　freopen은 stdin, stdout, stdout, stderr에
　　부착된 파일을 변경하는데 유용하다. mode스
　　트링에 대한 상세한 설명은 fopen을 참조한다

리 턴 값　에러없이 수행되면 인자 stream을 리턴하고

에러발생시 null을 리턴한다.
이 식 성 UNIX 시스템에상에서 사용할 수 있으며 ANSI-C와 호환된다.
참 조 `fclose`,`fdopen`,`fopen`,`open`,`setmode`

쌇fscanf
#함수 FSACNF

기 능 스트림으로부터 포맷된 입력을 수행한다.
구 문 #include <stdio.h>
int fscanf(FILE *stream, const char *format[,address, ...]);
프로토타입 `stdio.h`
기능 설명 한 번에 한 문자씩 스트림으로부터 읽어 오면서 일련의 입력 필드를 스캔한다.

format이 가리키는 포맷 문자열에서 fscanf에 전달된 포맷 지정에 따라 각 필드가 포맷된다 또한 fscanf는 format다음에 오는 인자로서 그것에 전달된 어드레스에 포맷된 입력을 저정한다. 이 때 입력 필드와 같은 수의 어드레스에 있어야 한다. 포맷지정에 관한 사항은 `scanf`를 참고한다.

리 턴 값 성공적으로 스캔되고 변환되어 저장된 입력 필드수를 리턴하는데, 이 리턴값은 저장되지 않고 스캔된 필드는 포함하지 않는다. fscanf가 파일종료(end-of-file)을 읽어들이는 경우 리턴값은 EOF가 된다.
지정된 필드가 없다면 리턴값은 0이다.
이 식 성 UNIX 시스템에상에서 사용할 수 있으며 ANSI-C와 호환된다.
참 조 `atof`,`cscanf`,`fprintf`,`printf`,`scanf`, `sscanf`,`vfscanf`,`vscanf`,`vsscanf`

쌇fseek
#함수 FSEEK

기 능 스트림상에 있는 파일 포인터의 위치를 변경시킨다. (파일 내의 입출력 위치 변경)
구 문 #include <stdio.h>

int seek(FILE *stream, long int offset,
int where);

프로토타입 `stdio.h`

기능 설명 stream과 연관된 파일 포인터에 where에 의해 주어진 파일 위치로부터 offset바이트 까지의 새로운 위치를 설정한다. 텍스크 모드 스트림에 대해서 offset은 0이 되거나 ftell에 의해서 리턴되는 값이어야 한다.

where	파일 위치
SEEK_SET(0)	파일의 시작 부분
SEEK_CUR(1)	현재 파일 포인터의 위치
SEEK_END(2)	파일의 끝

리 턴 값 포인터가 에러없이 리턴되면 0을 리턴하고 그렇지 못하면 0 이외의 값을 리턴한다.

이 식 성 UNIX 시스템에상에서 사용할 수 있으며 ANSI-C와 호환된다.

참 조 `fgetpos`,`fopen`,`fsetpos`,`ftell`, `lseek`,`rewind`,`setbuf`,`tell`

쌇fsetpos

#함수 FSETPOS

기 능 스트림의 파일 포인터 위치를 설정한다.

구 문 #include <stdio.h>
int fsetpos(FILE *stream,
const fpos_t *pos);

프로토타입 `stdio.h`

기능 설명 stream과 연관된 파일 포인터에 새로운 위치를 설정한다. 새로운 위치는 이전의 호출한 스트림상의 fgetpos에 의해 얻어진 값이다. 또한 그것은 stream이 가리키는 파일상의 파일 종료 표시를 초기화시켜 ungetc가 될 수 없게 한다. 다시 한 번 fserpos를 호출한 후에는 입출력이 가능하게 된다.

리 턴 값 에러없이 수행되면 0을 리턴한다. 에러가 발생 되면 0이외의 값이 리턴되며,`errno`에 0 이외의 값이 설정된다.

이 식 성 ANSI-C와 호환된다.

참　　조　`fgetpos`,`fseek`,`ftell`

쌿ftell
#함수 FTELL

기　　능　현재의 파일 포인터를 리턴한다.
구　　문　#include <stdio.h>
　　long int ftell(FILE *stream);
프로토타입　`stdio.h`
기능 설명　stream에 대한 현재의 파일 포인터를 리턴한
　　다. 파일의 처음부터 현재 위치의 오프셋을
　　바이트 단위로 측정하여 리턴한다. ftell에
　　의해 측정된 값은 fseek에서 사용된다.
리 턴 값　에러가 없으면 현재 파일 포인터의 위치를 리
　　턴한다. 에러시에는 -1L을 리턴한다.
이 식 성　UNIX 시스템에상에서 사용할 수 있으며 ANSI-
　　C와 호환된다.
참　　조　`fgetpos`,`fseek`,`fsetpos`,`lseek`,`tell`

쌿fwrite
#함수 FWRITE

기　　능　데이터를 스트림에 작성한다.
구　　문　#include <stdio.h>
　　size_t fwrite(const void *ptr,size_t size,
　　　size_t n,FILE *stream);
프로토타입　`stdio.h`
기능 설명　개별적인 size로 n개의 데이터를 지정된 출력
　　파일에 부가한다. 작성된 데이터는 ptr에서
　　시작된다.
　　작성된 전체 바이트수는 n * size가 된다.
　　ptr은 데이터 목적에 대한 포인터이다.
리 턴 값　에러없이 수행되면 fwrite는 실제로 작성된
　　항목수를 리턴하지만 에러가 발생하면 0이 리
　　턴된다.
이 식 성　UNIX 시스템에상에서 사용할 수 있으며 ANSI-
　　C와 호환된다.
참　　조　`fopen`,`fread`

쌿getc
#함수 GETC

기　　능　스트림으로부터 문자를 읽어들인다.
구　　문　#include <stdio.h>
　　int getc(FILE *stream);
프로토타입　`stdio.h`
기능 설명　주어진 입력 스트림에서 다음 문자를 리턴하고, 그 다음문자를 가리키도록 파일 포인터를 이동한다.
리 턴 값　에러없이 수행되면 읽어들인 문자를 부호없는 int로 변환한 다음 리턴한다. 파일 종료나 에러시에는 EOF를 리턴한다.
이 식 성　UNIX 시스템상에서 사용할 수 있으며 ANSI-C와 호환된다.
참　　조　`fgetc`,`getch`,`getchar`,`getche`,`eets`,`putc`,`putchar`,`ungetc`

쌿getchar
#함수 GETCHAR

기　　능　stdin으로부터 문자를 읽어들인다.
구　　문　#include <stdio.h>
　　int getchar(void);
프로토타입　`stdio.h`
기능 설명　명명된 입력 스트림 stdin상에서 다음문자를 리턴하는 매크로이다. 이것은 getc(stdin)와 같의 의미로 정의된다.
리 턴 값　에러가 없으면 getchar은 읽어들인 문자를 부호없는 int로 변환시킨 후에 리턴한다. 파일 종료시나 에러시에는 EOF를 리턴한다.
이 식 성　UNIX 시스템상에서 사용할 수 있으며 ANSI-C와 호환된다.
참　　조　`fgetc`,`fgetchar`,`getc`,`getch`,`getche`,`putc`,`putchar`,`ungetc`

쌿gets
#함수 GETS

기　　능　stdin으로부터 문자열을 읽어 들인다.
구　　문　char *gets(char *s);
프로토타입　`stdio.h`
기능 설명　표준 입력 스트림 stdin으로부터 1문자를 읽

어 들이고 문자형 포인터 s가 가리키는 장소
에 저장한다. 다음 행을 캐리지 리턴(carrig
e return)할 때까지의 문자열을 읽어들인 후
저장될 때에 null문자로 변환한다.
리 턴 값 에러없이 수행되면 문자열 인수 *s를 리턴하
고 EOF나 에러발생시에 null을 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있으며 ANSI-C
와 호환된다.
참 조 `cgets`,`ferror`,`fgets`,`fputs`,`getc`,
`puts`

쌇getw
#함수 GETW

기 능 스트림으로부터 정수를 읽어온다.
구 문 #include <stdio.h>
int getw(FILE *stream);
프로토타입 `stdio.h`
기능 설명 명명된 입력 스트림에서 다음 정수를 리턴한
다. 파일의 정수값에 특별한 서식은 가정하지
않는다. getw는 텍스트 모드에서 스트림이 개
방되어 있을 때는 사용할 수가 없다.
리 턴 값 입력 스트림상에서 다음 정수를 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있다.
참 조 `putw`

쌇perror
#함수 PERROR

기 능 시스템 에러메시지를 리턴한다.
구 문 void perror(const char *s);
프로토타입 `stdio.h`
기능 설명 에러를 발생한 라이브러리 루틴에 대한 시스
템 에러메시지를 stderr스트림에 인쇄한다.
먼저, 인수 s가 인쇄되고 나서 콜론, 그리고
errno의 현재값에 대응하는 메시지가 인쇄되
며 마지막으로 새로운 행이 인쇄된다.
리 턴 값 없음
이 식 성 UNIX 시스템상에서 사용할 수 있으며 ANSI-C
와 호환된다.
참 조 `clearerr`,`eof`,`_strerror`,`strerror`

쌇printf
#함수 PRINTF

기　　능　포맷된 출력을 stdout으로 보낸다.
구　　문　int printf(const char *format
　　[,argument, ...]);
프로토타입　`stdio.h`
기능 설명　일련의 인수를 받아서 format에 의해 주어진 포맷 문자열 형식 지정에 따라 포맷된 데이터를 stdout으로 출력한다. 여기서는 반드시 인수의 수와 같은 수의 형식지정이 있어야 한다

포맷 문자열은 ..printf 함수 호출 안에서 주어지는 것으로 그 인수를 변환하고 형식을 지정하며 인쇄하는 방법을 지정한다.
이 때 포맷에 대한 인수는 충분히 있어야 한다. 그렇지 않으면 예상치 못한 결과가 나오게 된다.

* 보통 문자는 인수의 값을 사용하지 않고 그대로 출력 스트림에 복사한다.
* 변환 지정은 인수 리스트로부터 인수를 불러 와서 이들에 포맷을 적용시킨다.

포맷 지정

printf 포맷 지정은 다음과 같은 형태를 갖는다.
% [flags] [width] [.prec] [F|N|H|I|L] type
각각의 변환지정은 퍼센트 문자(%)로 시작된다. % 뒤에는 다음과 같은 순서대로 온다.

* 임의의 플래그 문자의 연속 [flags]
* 임의의 필드폭 지정자 [width]
* 임의의 정밀도 지정자 [.prec]
* 임의의 입력 크기 수정자 [F|N|H|I|L]
* 변환형 문자 [type]

리 턴 값　출력된 바이트수를 리턴한다. 에러시에는 EOF를 리턴한다.
이 식 성　UNIX 시스템상에서 사용할 수 있으며 ANSI-C

와 호환된다.
참 조 `cprintf`,`ecvt`,`fprintf`,`fread`,
`fscanf`,`putc`,`putw`,`scanf`,`speintf`,
`vprintf`,`vsprintf`

쌿putc
#함수 PUTC

기 능 스트림에 1문자를 출력한다.
구 문 #include <stdio.h>
int putc(int c, FILE *stream);
프로토타입 `stdio.h`
기능 설명 stream에 의해 주어진 스트림에 문자 c를 출력하는 매크로이다.
리 턴 값 에러가 없으면 putc는 출력되는 문자 c를 리턴한다. 에러가 발생하면 EOF를 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있으며 ANSI-C와 호환된다.
참 조 `fprintf`,`fputc`,`fputch`,`getc`,
`getchar`,`printf`,`putch`,`putchar`

쌿putchar
#함수 PUTCHAR

기 능 stdout상에 문자를 출력한다.
구 문 #include <stdio.h>
int putchar(int c);
프로토타입 `stdio.h`
기능 설명 putchar(c)는 putc(c,stdout)로 정의된 매크로이다.
리 턴 값 에러가 없으면 문자 c를 리턴하고, 에러 발생시 EOF를 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있으며 ANSI-C와 호환된다.
참 조 `fputchar`,`getc`,`getchar`,`putc`,`putch`,
`puts`

쌿puts
#함수 PUTS

기 능 stdout 스트림으로 문자열을 출력한다.

구　　문　int puts(const char *s);
프로토타입　`stdio.h`
기능 설명　null 종료 문자열 s를 표준 출력 스트림 stdout로 복사하고 뒤에 행의 문자를 추가한다.
리 턴 값　에러가 없으면 puts는 음이 아닌 값을 리턴하고, 에러 발생시 EOF가 리턴된다.
이 식 성　UNIX 시스템상에서 사용할 수 있으며 ANSI-C와 호환된다.
참　　조　`cputs`,`fputs`,`gets`,`printf`,`putchar`

쌇putw
#함수 PUTW

기　　능　스트림상에 정수를 출력한다.
구　　문　#include <stdio.h>
int putw(int w, FILE *stream);
프로토타입　`stdio.h`
기능 설명　주어진 스트림에 정수 w를 출력한다. putw는 파일에서 특별한 정렬을 시키거나 요구하지 않는다.
리 턴 값　에러가 없으면 putw는 정수 w를 리턴하고 에러 발생시 EOF를 리터한다. EOF는 옳바른 정수이기 때문에 putw로 에러를 탐지하려면 ferror이 사용되어야 한다.
이 식 성　UNIX 시스템상에서 사용될 수 있다.
참　　조　`getw`,`printf`

쌇remove
#함수 REMOVE

기　　능　파일을 삭제한다.
구　　문　#include <stdio.h>
int remove(const char *filename);
프로토타입　`stdio.h`
기능 설명　filename으로 지정된 파일을 삭제한다. 이 함수는 단순히 unlink를 호출하는 함수 매크로이다.
리 턴 값　에러가 없으면 remove는 0을 리턴한다. 에러 발생시 -1을 리턴하고`errno`는 아래값 중 하나가 된다.

ENOENT 해당 파일이나 디렉토리가 없다.
EACESS 억세스 허용이 않된다.
이 식 성 UNIX 시스템상에서 사용할 수 있으며 ANSI-C 와 호환된다.
참 조 `unlink`

쌇rename
#함수 RENAME

기 능 파일의 이름을 변경한다.
구 문 #int rename(const char *oldname,
const char *newname);
프로토타입 `stdio.h`
기능 설명 파일의 이름을 oldname에서 newname으로 변경한다. 만일 드라이브 이름이 newname에 있는 경우 그 이름은 oldname 드라이브와 일치하여야 한다.

oldname의 디렉토리와 newname의 디렉토리가 일치하여야 하는 것은 아니다. rename은 디렉토리간의 파일 이동 에도 사용할 수 있다.
리 턴 값 에러가 없으면 rename은 0을 리턴한다. 에러 발생시 -1을 리턴하고`errno`는 아래의 값 중 하나로 설정된다.

ENOENT 해당 파일이나 디렉토리가 없다.
EACESS 억세스 허용이 않된다.
ENOTSAM 같은 디바이스가 아니다.
이 식 성 ANSI-C와 호환된다.

쌇rewind
#함수 REWIND

기 능 스트림의 앞부분에 파일 포인터를 위치시킨다
구 문 #include <stdio.h>
void rewind(FILE *stream);
프로토타입 `stdio.h`
기능 설명 rewind(stream)은 fseek가 EOF마을 초기화시키는데 비해서 EOF와 에러표시를 모두 지운다는 점을 제외하면 fseek(stream,0L,SEEK_SET)을 호출하는것과 같다.

rewind이후에 파일 갱신에 대한 조작은 입/출력이 모두 가능하다.

리 턴 값 없음

이 식 성 UNIX 시스템상에서 사용할 수 있으며 ANSI-C와 호환된다.

참 조 `fopen`,`fseek`,`ftell`

쌇scanf

#함수 SCANF

기 능 stdin 스트림으로부터 입력을 검색하고 포맷한다.

구 문 int scanf(const char *format[,address,...]);

프로토타입 `stdio.h`

기능 설명 stdin으로부터 한 번에 한 문자씩 읽고 검색한 후 format이 지정한 문자열에 들어 있는 포맷 지정에 따라 내용을 변경하여 인수로 주어진 어드레스에 내용을 넣는다.

포맷 지정

...scanf 의 포맷은 아래와 같은 형태를 가진다.

[*] [width] [F|N] [H|I|L] type_character

* 임의의 지정 금지 문자 [*]
* 임의의 폭 지정자 [width]
* 임의의 포인터 크기 수정자 [FILE]
* 임의의 인수 유형 수정자 [H|I|L]
* 형문자 (type_character)

리 턴 값 에러없이 스캔, 변환, 포맷 지정된 입력 필드의 수 를 리턴한다. 단지 스캔만 되고 변환되지 않은 필드 수는 리턴값에 포함되지 않는다 scanf가 파일의 끝을 읽으려는 경우, EOF를 리턴한다. 저장된 필드가 없으면 0이 리턴된다.

이 식 성 UNIX 시스템상에서 사용할 수 있으며 ANSI-C와 호환된다.

참 조 `cscanf`,`fscanf`,`printf`,`sscanf`,

`cfscanf`,`vscanf`,`vsscanf`

쌇setbuf
#함수 SETBUF

기　　능　스트림에 대한 버퍼를 지정한다.
구　　문　#include <stdio.h>
　　void setbuf(FILE *stream, char *buf);
프로토타입　`stdio.h`
기능 설명　I/O를 위해 자동적으로 할당된 버퍼 대신에 buf로 주어진 버퍼를 사용하도록 한다. 이것은 스트림이 열린 이후에 사용한다.

　　만일 buf가 null이면 I/O는 버퍼링을 하지 않는다. 주어지는 버퍼의 크기는 BUFSIZ(stdio.h) 이상의 크기를 가져야 한다.

　　stdin과 stdout은 방향이 전환되지 않는다면 버퍼되지 않으며, 그 반대는 완전히 버퍼된다 setbuf는 버퍼링 여부를 바꾸기 위해서 사용될 수 있다.
리 턴 값　없음
이 식 성　UNIX 시스템상에서 사용할 수 있으며 ANSI-C와 호환된다.
참　　조　`fflush`,`fopen`,`fseek`,`setvbuf`

쌇setvbuf
#함수 SETVBUF

기　　능　버퍼링을 스트림에 지정한다.
구　　문　#include <stdio.h>
　　void setvbuf(FILE *stream, char *buf,
　　int type,size_t size);
프로토타입　`stdio.h`
기능 설명　I/O 버퍼링을 위해 자동적으로 할당된 버퍼 대신에 buf로 주어진 버퍼를 사용하도록 한다 이것은 stream이 열린 이후에 사용한다.
　　만일 buf가 null이면`malloc`으로 size만큼의 메모리를 할당한다. size는 버퍼의 크기를 나타내며 0보다 커야한다.
리 턴 값　에러가 없으면 0을 리턴한다. type나 size의

값이 부적당하거나, malloc할 메모리가 부족
하면 0이외의 값을 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있으며 ANSI-C
와 호환된다.
참 조 `fflush`,`fopen`,`setbuf`

쌇sprintf
#함수 SPRINTF

기 능 포맷된 출력을 쓴다.
구 문 int sprinf(char *buffer,
const char *format [,argument, ...]);
프로토타입 `stdio.h`
기능 설명 일련의 인수를 받아서 format에 지정된 포맷
의 결과를 buffer로 지정된 문자열에 기록한
다.

sprintf는 포맷 지정과 인수를 순서대로 지정
한다. 따라서 포맷 지정의 수와 인수의 수는
일치하여야 한다.
리 턴 값 출력 바이트수를 리턴한다. 종료를 나타내는
null은 계산에 포함되지 않는다. 에러발생시
sprintf는 EOF를 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있으며 ANSI-C
와 호환된다.
참 조 `fprintf`,`printf`

쌇sscanf
#함수 SSCANF

기 능 문자열로부터 입력을 스캔하고 포맷한다.
구 문 int sscanf(const char *buffer,
const char *format [,address, ...]);
프로토타입 `stdio.h`
기능 설명 문자열로부터 읽은 입력 필드를 한 번에 한
문자씩 조사한다. 각 필드는 format이 지정한
포맷 문자열에 있는 sscanf에 전달된 포맷 지
정에 따라 형식화된다.
리 턴 값 에러없이 스캔, 변환, 포맷 지정된 입력 필드
의 수 를 리턴한다. 단지 스캔만 되고 변환되
지 않은 필드 수는 리턴값에 포함되지 않는다

scanf가 파일의 끝을 읽으려는 경우, EOF를 리턴한다. 저장된 필드가 없으면 0이 리턴된다.

이 식 성 UNIX 시스템상에서 사용할 수 있으며 ANSI-C와 호환된다.

참 조 `fscanf`,`scanf`

쌇_strerror

#함수 _STRERROR

기 능 에러메시지 문자열의 포인터를 리턴한다.

구 문 char *strerror(const char *s);

프로토타입 `string.h`,`stdio.h`

기능 설명 사용자에 맞는 에러메시지를 설정할 수 있도록 한다. 에러 메시지를 포함하는 null 종료 문자열의 포인터를 리턴한다.

리 턴 값 구성된 에러 메시지에 대한 포인터를 리터한다. 포인터가 가리키는 내용은 정적 버퍼로서 _strerror을 호출할 때 마다 이전의 내용은 지워진다.

이 식 성 ANSI-C와 호환된다.

참 조 `perror`,`strerror`

쌇strerror

#함수 STRERROR

기 능 에러 메시지 문자열의 포인터를 리턴한다.

구 문 char *strerror(int errnum);

프로토타입 `string.h`,`stdio.h`

기능 설명 에러 번호, int 파라미터 errnum을 받아서 errnum과 관계가 있는 에러 메시지 문자열에 대한 포인터를 리턴한다.

리 턴 값 구성된 에러 메시지에 대한 문자열의 포인터를 리턴한다. 포인터가 가리키는 내용은 정적 버퍼로서 strerror을 호출할 때 마다 이전의 내용은 지워진다.

이 식 성 ANSI-C와 호환된다.

참 조 `perror`,`_strerror`

쌇tmpfile

#함수 TMPFILE

기　　능　임시 파일을 이진 모드로 연다.
구　　문　char *tmpfile(void);
프로토타입　`stdio.h`
기능 설명　임시 이잔 파일을 생상하고 갱신이 가능하도록 오픈한다. 이 파일이 닫히게 되거나 프로그램이 종료되면 파일은 자동적으로 삭제된다.
리 턴 값　임시로 열린 스트림 파일의 포인터를 리턴한다. 만일 파일을 생성할 수 없으면 null을 리턴한다.
이 식 성　UNIX 시스템상에서 사용할 수 있으며 ANSI-C와 호환된다.

쌇tmpnam
#함수 tmpnam

기　　능　독특한 파일 이름을 작성한다.
구　　문　char *tmpfile(char *s);
프로토타입　`stdio.h`
기능 설명　임시 파일명으로 쓸 수 있는 독특한 파일명을 만든다. 사용자가 호출할 때마다 서로 다른 이름이 만들어지는 데 호출 한도는 stdio.h에서 정의되어 있는 TMP_MAX로 65535 이다.

만일 사용자가 tmpnam을 이용하여 만들어 놓은 임시파일이 있다면 이는 자동으로 삭제되지 않으므로, remove등을 이용하여 입시 파일이 필요없을 때 지워 버린다.
리 턴 값　만일 s가 null이면 tmpnam은 자신 내부에 있는 정적 배열에 대한 포인터를 리턴하며, 그 밖에는 s를 리턴한다.
이 식 성　UNIX 시스템상에서 사용할 수 있으며 ANSI-C와 호환된다.
쌇ungetc
#함수 UNGETC

기　　능　입력 스트림에 한 문자를 되돌린다.
구　　문　#include <stdio.h>
int ungetc(int c, FILE *stream);
프로토타입　`stdio.h`

기능 설명 stream으로 이름을 가진 입력에 문자 c를 되돌린다. 그 스트림은 입력모드로 열려 있어야 한다. 이렇게 돌려진 문자는 다음의`getc`혹은`fread`에 의해서 리턴된다. 하나의 문자는 언제라도 되돌려질 수 있다.

리 턴 값 에러가 없으면 c를 리턴하고, 에러가 발생하면 EOF를 리턴한다.

이 식 성 UNIX 시스템상에서 사용할 수 있으며 ANSI-C와 호환된다.

참 조 `fgetc`,`getc`,`getchar`

쌇vfprintf

#함수 VFPRINTF

기 능 스트림에 포맷됭 출력을 기록한다.

구 문 #include <stdio.h>
int vfprintf(FILE *stream,
const char *format,va_list arglist);

프로토타입 `stdio.h`

기능 설명 v...printf 함수는 ...printf함수 계열로 알려져있다. ...printf함수와 동일한 작동을 하지만, 인수 리스트 대신 인수의 포인터를 받아들인다.

vfprintf는 일련의 인수 리스트 포인터를 받아서 format에 지정된 문자열 포맷을 각각의 인수에 적용하여 스트림에 포맷 데이터를 출력한다. 지정된 포맷의 수와 인수의 수는 일치 하여야 한다.

리 턴 값 출력된 바이트 수 를 리턴한다. 에러 발생시 EOF를 리턴한다.

이 식 성 UNIX 시스템 V에서 사용할 수 있으며 ANSI-C와도 호환된다.

참 조 `printf`

쌇vfscanf

#함수 VFSCANF

기 능 스트림으로부터 입력을 스캔하고 포맷한다.

구 문 #include <stdio.h>
int vfscanf(FILE *stream,

const char *format, va_lst arglist);
프로토타입 `stdio.h`
기능 설명 v..scanf 함수는 ..scanf 계열의 함수로 알려져있다. ..scanf함수와 동일한 작동을 하며, 인수 리스트 대신 인수 리스트의 포인터를 받아들인다.
리 턴 값 에러없이 스캔, 변환, 포맷 지정된 입력 필드의 수 를 리턴한다. 단지 스캔만 되고 변환되지 않은 필드 수는 리턴값에 포함되지 않는다 scanf가 파일의 끝을 읽으려는 경우, EOF를 리턴한다. 저장된 필드가 없으면 0이 리턴된다.
이 식 성 UNIX 시스템 V에서 사용할 수 있다.
참 조 `fscanf`,`scanf`

쌿vprintf
#함수 VPRINTF

기 능 포맷된 출력을 stdout에 기록한다.
구 문 #include <stdarg.h>
int vprintf(const char *format,
va_lst arglist);
프로토타입 `stdio.h`
기능 설명 v...printf 함수는 ...printf함수 계열로 알려져있다. ...printf함수와 동일한 작동을 하지만, 인수 리스트 대신 인수의 포인터를 받아들인다.

vprintf는 일련의 인수 리스트 포인터를 받아서 format에 지정된 문자열 포맷을 각각의 인수에 적용하여 스트림에 포맷 데이터를 출력한다. 지정된 포맷의 수와 인수의 수는 일치 하여야 한다.
리 턴 값 출력된 바이트 수 를 리턴한다. 에러 발생시 EOF를 리턴한다.
이 식 성 UNIX 시스템 V에서 사용할 수 있으며 ANSI-C와도 호환된다.
참 조 `printf`

쌿vscanf
#함수 VSCANF

기 능 stdin으로부터 입력을 스캔하고 포맷한다.
구 문 #include <stdio.h>
int vscanf(
const char *format, va_lst arglist);
프로토타입 `stdio.h`
기능 설명 v..scanf 함수는 ..scanf 계열의 함수로 알려져있다. ..scanf함수와 동일한 작동을 하며, 인수 리스트 대신 인수 리스트의 포인터를 받아들인다.
리 턴 값 에러없이 스캔, 변환, 포맷 지정된 입력 필드의 수 를 리턴한다. 단지 스캔만 되고 변환되지 않은 필드 수는 리턴값에 포함되지 않는다 vscanf가 파일의 끝을 읽으려는 경우, EOF를 리턴한다. 저장된 필드가 없으면 0이 리턴된다.
이 식 성 UNIX 시스템 V에서 사용할 수 있다.
참 조 `fscanf`,`scanf`

쌇vsprintf
#함수 VSPRINTF

기 능 포맷된 출력을 stdout에 기록한다.
구 문 #include <stdarg.h>
int vsprintf(char *buffer,const char
*format, va_lst arglist);
프로토타입 `stdio.h`
기능 설명 v...printf 함수는 ...printf함수 계열로 알려져있다. ...printf함수와 동일한 작동을 하지만, 인수 리스트 대신 인수의 포인터를 받아들인다.

vsprintf는 일련의 인수 리스트 포인터를 받아서 format에 지정된 문자열 포맷을 각각의 인수에 적용하여 스트림에 포맷 데이터를 출력한다. 지정된 포맷의 수와 인수의 수는 일치 하여야 한다.
리 턴 값 출력된 바이트 수 를 리턴한다. 에러 발생시 EOF를 리턴한다.
이 식 성 UNIX 시스템 V에서 사용할 수 있으며 ANSI-C와도 호환된다.

참 조 `printf`

쌇vsscanf
#함수 VSSCANF

기 능 문자열로부터 입력을 스캔하고 포맷한다.
구 문 #include <stdio.h>
int vscanf(char *buffer,
const char *format, va_lst arglist);
프로토타입 `stdio.h`
기능 설명 v..scanf 함수는 ..scanf 계열의 함수로 알려져있다. ..scanf함수와 동일한 작동을 하며, 인수 리스트 대신 인수 리스트의 포인터를 받아들인다.
리 턴 값 에러없이 스캔, 변환, 포맷 지정된 입력 필드의 수 를 리턴한다. 단지 스캔만 되고 변환되지 않은 필드 수는 리턴값에 포함되지 않는다 vsscanf가 파일의 끝을 읽으려는 경우, EOF를 리턴한다. 저장된 필드가 없으면 0이 리턴된다.
이 식 성 UNIX 시스템 V에서 사용할 수 있다.
참 조 `fscanf`,`scanf`

쌇stdlib.h
#헤더파일 STDLIB.H

함수

`abort` `abs` `atexit`
`atof` `atoi` `atol`
`bsearch` `calloc` `div`
`ecvt` `_exit` `exit`
`fcvt` `free` `gcvt`
`getenv` `itoa` `labs`
`ldiv` `lfind` `_lrotl`
`_lrotr` `lsearch` `ltoa`
`malloc` `max` `min`
`putenv` `qsort` `rand`
`random` `randomize` `realloc`
`_rotl` `_rotr` `srand`
`strtod` `strtol` `strtoul`

`swab` `system`
`ultoa`

쌇atexit
#함수 ATEXIT

기 능 종료 함수를 레지스터한다.
구 문 #include <stdlib.h>
int atexit(atexit_t func);
프로토타입 `stdlib`
기능 설명 exit함수로서, func에 의해 지정된 함수를 레지스터 한다. 프로그램이 정상적으로 종료하는 경우, exit는 운영체제로 리턴하기 바로 전에 (*func)()를 호출한다. 호출된 함수는 atexit_t형이고, 이 함수는 stdlib.h에서 typedef로 정의된다.

atexit 호출은 다른 exit 함수를 레지스터 한다. 함수 func는 32개 까지 등록할 수 있으며 이들 함수는 후입 선출식(last-in, first-out)으로 실행된다. 즉 나중에 레지스터된 함수가 먼저 실행된다.
리 턴 값 에러가 없으면 0을 리턴하고 에러가 발생하면 0이외의 값을 리턴한다.
이 식 성 ANSI-C에서 사용할 수 있다.
참 조 `abort`,`_exit`,`exit`,`spawn`

쌇atoi
#함수 ATOI

기 능 문자열을 정수로 변환한다.
구 문 int atoi(const char *s);
프로토타입 `stdlib.h`
기능 설명 s에 의하여 지정된 문자열을 int로 변환한다. atoi의 인식은 다음의 명령에 의한다.

* 탭과 공백이 있는 임의의 문자열
* 임의의 부호
* 디지트 문자열

리 턴 값 입력 문자의 변환된 값을 리턴한다. 믄자열에

대응하는 형(int)의 수로 변환될 수 없다면,
리턴값은 0이다.
이 식 성 UNIX 시스템 V에서 사용할 수 있으며 ANSI-C
와도 호환된다.
참 조 `atof`,`atol`,`ecvt`,`fcvt`,`gcvt`

쌇atol
#함수 atol

기 능 문자열을 long으로 변환한다.
구 문 long atol(const char *s);
프로토타입 `stdlib.h`
기능 설명 s에 의하여 지정된 문자열을 long으로 변환한
다. atol의 인식은 다음의 명령에 의한다.

* 탭과 공백이 있는 임의의 문자열
* 임의의 부호
* 디지트 문자열

리 턴 값 입력 문자의 변환된 값을 리턴한다. 믄자열에
대응하는 형(long)의 수로 변환될 수 없다면,
리턴값은 0이다.
이 식 성 UNIX 시스템 V에서 사용할 수 있으며 ANSI-C
와도 호환된다.
참 조 `atof`,`atoi`,`ecvt`,`fcvt`,`gcvt`

쌇bsearch
#함수 BSEARCH

기 능 배열의 이진 탐색
구 문 #include <stdlib.h>
void *bsearch(const void *key,void *base,
size_t nelem, size_t width,
int (*temp)(const void*,const void*));
프로토타입 `stdlib.h`
기능 설명 메모리 내에서 nelem 원소의 테이블(배열)을
탐색하고 탐색키와 일치하는 첫번째 요소(en
try)의 어드레스를 리턴한다.
일치되는 것이 없을 때에는 0을 리턴한다.
size_t는 unsigned int로 정의된다.
리 턴 값 탐색 key와 일치하는 테이블 내의 첫번째 요

소의 어드레스를 리턴한다.
이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C 와 도 호환된다.
참 조 `lfind`,`lsearch`,`qsort`

쌿div
#함수 DIV

기 능 두 개의 정수를 나눈 후 몫과 나머지를 리턴 한다.
구 문 #include <stdlib.h>
div_t div(int numer, int denom);
프로토타입 `stdlib.h`
기능 설명 두 정수를 나눈 다음 div_t 형으로 몫과 나머 지를 리턴한다.
리 턴 값 원소 quot(몫)와 rem(나머지)을 갖는 구조체 를 리턴한다.
이 식 성 ANSI-C와 호환된다.
참 조 `ldiv`

쌿ecvt
#함수 ECVT

기 능 부동 소숫점 수를 문자열로 변환한다.
구 문 char *ecvt(double value,int ndig,
int *dec, int *sign);
프로토타입 `stdlib.h`
기능 설명 value를 맨 위쪽의 부호 digit으로 시작되는 디지트의 null종료 문자열로 변환하고, 문자 열에 대한 포인터를 리턴한다. 정수 dec에는 소숫점으로부터의 위치가 대입되어 있다.
리 턴 값 ecvt의 리턴값은 매번 ecvt 호출에 의해 고쳐 진 내용을 갖는 정적 데이타를 가리킨다.
이 식 성 UNIX 시스템상에서 사용할 수 있다.
참 조 `atof`,`atoi`,`atol`,`fcvt`,`gcvt`,`printf`

쌿fcvt
#함수 FCVT

기 능 부동 소숫점 수를 문자열로 변환한다.
구 문 char *fcvt(double value,int ndig,

int *dec, int *sign);
프로토타입 `stdlib.h`
기능 설명 value를 맨 위쪽의 부호 digit으로 시작되는 디지트의 null종료 문자열로 변환하고, 문자열에 대한 포인터를 리턴한다. 정수 dec에는 소숫점으로부터의 위치가 대입되어 있다.
리 턴 값 fcvt의 리턴값은 매번 fcvt 호출에 의해 고쳐진 내용을 갖는 정적 데이타를 가리킨다.
이 식 성 UNIX 시스템상에서 사용할 수 있다.
참 조 `atof`,`atoi`,`atol`,`ecvt`,`gcvt`,`printf`

쌇gcvt
#함수 GCVT

기 능 부동 소숫점 수를 문자열로 변환한다.
구 문 char *gcvt(double value,int ndec, char *buf);
프로토타입 `stdlib.h`
기능 설명 value를 null로 종료되는 ASCII문자열로 변환하고 그 문자열은 buf에 저장한다. 가능하면 fortran의 F형식으로 하여금 ndec 유효자릿수를 작성하거나 printf의 E-형식(프린트를 위해 준비)으로 값을 리턴한다. 뒤에 계속되는 0은 무시된다.
리 턴 값 buf가 가리키는 문자열의 어드레스를 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있다.
참 조 `ecvt`,`fcvt`

쌇getenv
#함수 GETENV

기 능 환경변수의 문자열을 구한다.
구 문 char *getenv(const char *name);
프로토타입 `stdlib.h`
기능 설명 지정된 변수값을 리턴한다. 이 변수명은 등호기호(=)를 제외한 소문자나 대문자로 할 수 있다. 만일 지정된 환경변수가 없을 때 에는 빈 문자열을 리턴한다.
리 턴 값 에러가 없으면 연관된 값을 리턴한다. 지정된 name이 환경에서 정의되어 있지 않은경우 빈

문자열을 리턴한다. 환경을 변경하고자 할 경우에는`putenv`를 사용한다.

이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C 와도 호환된다.

참 조 `getpsp`,`putenv`

쌇itoa

#함수 ITOA

기 능 정수를 문자열로 변환한다.

구 문 char *itoa(int value, char *string, int radix);

프로토타입 `stdlib.h`

기능 설명 이 함수는 주어진 정수치 value를 null종료 문자열로 변환하는 함수이다. radix는 value를 변환하는 데 필요한 밑수를 지정하는데 이 밑수는 2에서 32까지의 숫자이다. value가 음수이고 radix가 10인 경우에 string의 첫번째 문자는 마이너스(-) 부호가 된다.

리 턴 값 string을 가리키는 문자열을 리턴한다. 에러 리턴은 없다.

이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C 와도 호환된다.

참 조 `ltoa`,`ultoa`

쌇ldiv

#함수 LDIV

기 능 두 개의 long형을 나누고 그 몫과 나머지를 구한다.

구 문 #include <stdlib.h>
ldiv_t ldiv(long int number, long int denom);

프로토타입 `stdlib.h`

기능 설명 두 개의 long 형을 나누고 그 몫과 나머지를 ldiv_t 형으로서 리턴한다. number와 denum은 각각 분자와 분모이다.

리 턴 값 몫인 quot와 나머지인 rem의 요소를 갖는 구조체를 리턴한다.

이 식 성 ANSI-C와 호환된다.

참　　조　`div`

쌿lfind
#함수 LFIND

기　　능　선형 탐색을 수행한다.
구　　문　#include <stdlib.h>
　　void *lfind(const void *ley,
　　const void *base,size_t *num,
　　size_t width, int (*fcmp)
　　(const void*,const *void));
프로토타입　`stdlib.h`
기능 설명　순차적인 레코드 배열에서 key 값에 대한 선형 탐색을 한다. 또한 이것은 사용자가 정의한 비교루틴(fcmp)를 사용한다. 이 배열은 width바이트 크기의 *num 레코드를 가지고 있으며 base가 가리키는 메모리 위치에서 시작한다.
리 턴 값　탐색키와 일치하는 테이블 내에서의 첫번째 요소의 어드레스를 리턴한다. 일치하는것이 없을 경우 lfind는 null을 리턴한다.
참　　조　`bsearch`,`lsearch`

쌿_lrotl
#함수 _LROTL

기　　능　부호없는 long 형의 값을 좌측으로 회전(rotate) 시킨다.
구　　문　unsigned long _lrotl(unsiged log val,
　　int count);
프로토타입　`stdlib.h`
기능 설명　주어진 부호 없는 long 형 값 val을 count 비트 만큼 좌측으로 회전 시킨다.
리 턴 값　count 비트만큼 좌로 회전된 val의 값을 리턴한다.
참　　조　`_rotr`

쌿_lrotr
#함수 _LROTR

기　　능　부호없는 long 형의 값을 우측으로 회전(rota

te) 시킨다.

구 문 unsigned long _lrotl(unsiged log val, int count);

프로토타입 `stdlib.h`

기능 설명 주어진 부호 없는 long 형 값 val을 count 비트 만큼 우측으로 회전 시킨다.

리 턴 값 count 비트만큼 우로 회전된 val의 값을 리턴한다.

참 조 `_rotr`

쌇lsearch

#함수 LSEARCH

기 능 선형 탐색을 수행한다.

구 문 #include <stdlib.h>
void *lsearch(const void *ley,
const void *base,size_t *num,
size_t width, int (*fcmp)
(const void*,const *void));

프로토타입 `stdlib.h`

기능 설명 정보를 위해서 테이블을 탐색한다. 이는 선형 탐색이므로 테이블 요소는 lsearch 호출전에 분류될 필요는 없다. 만일 key가 가리키는 요소가 테이블 내에 없는 경우 lsearch는 테이블에 이 데이터를 부가시킨다.

* base는 탐색 테이블의 0번째 원소를 가리키는 포인터를 나타낸다.
* num은 테이블 요소(entry)의 정수 포인터를 나타낸다.
* width는 각 요소의 바이트 수를 나타낸다.
* key는 search key로서 탐색될 데이터를 가리키는 포인터를 나타낸다.

리 턴 값 테이블 내에서 탐색키와 일치하는 첫번째 요소의 어드레스를 리턴한다.

이 식 성 UNIX 시스템상에서 사용할 수 있다.

참 조 `bsearch`,`lfind`,`qsort`

쌇ltoa

#함수 LTOA

기 능 long형 정수를 문자열로 변환한다.
구 문 char *ltoa(int value, char *string,
int radix);
프로토타입 `stdlib.h`
기능 설명 이 함수는 주어진 정수치 long형 value를 n
ull종료 문자열로 변환하는 함수이다. radi
x는 value를 변환하는 데 필요한 밑수를 지
정하는데 이 밑수는 2에서 32까지의 숫자이
다. value가 음수이고 radix가 10인 경우에
string의 첫번째 문자는 마이너스(-) 부호가
된다.
리 턴 값 string을 가리키는 문자열을 리턴한다. 에러
리턴은 없다.
이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C 와
도 호환된다.
참 조 `itoa`,`ultoa`

쌿max
#함수 MAX

기 능 두 개의 값중 큰 값을 리턴한다.
구 문 #include <stdlib.h>
(type) max(a,b);
프로토타입 `stdlib.h`
기능 설명 이 매크로는 두 개의 값을 비교하고 그 중 큰
값을 리턴한다. 두 개의 인수와 함수 선언은
같은 형태이어야 한다.
리 턴 값 두개의 값중 큰 값을 리턴한다.

쌿min
#함수 min

기 능 두 개의 값중 작은 값을 리턴한다.
구 문 #include <stdlib.h>
(type) min(a,b);
프로토타입 `stdlib.h`
기능 설명 이 매크로는 두 개의 값을 비교하고 그 중 작
은값을 리턴한다. 두 개의 인수와 함수 선언
은 같은 형태이어야 한다.
리 턴 값 두개의 값중 작은 값을 리턴한다.

쌇putenv
#함수 putenv

기　　능　현재의 환경을 위한 문자열을 추가한다.
구　　문　int putenv(const char *name);
프로토타입　`stdlib.h`
기능 설명　문자열 name을 받아서 current 프로세스의 환경에 등록한다. 예를 들면
putenv("PATH=C:\FOOL");

putenv는 변수를 변경 또는 삭제하는데도 사용 할 수 있다. 변수 값에 아무것도 쓰지 않으면 (예,MYVAR=)현재 등록된 것이 삭제된다 또한 putenv는 현재의 환경을 변경하는 데만 사용할 수 있다. 프로그램이 종료되면 원래의 환경으로 복귀한다.
리 턴 값　에러가 없으면 0을 리턴하고, 에러시에는 -1을 리턴한다.
이 식 성　UNIX 시스템상에서 사용할 수 있다.
참　　조　`getenv`

쌇qsort
#함수 QSORT

기　　능　주어진 자료를 고속 소트 알고리즘을 이용하여 정렬한다.
구　　문　void qsort(void *base, size_t nelem,
size_t width, int (*fcmp)
(const *void,const *void);
프로토타입　`stdlib.h`
기능 설명　고속 소트 알고리즘의 변형인 "medium of three" 를 실행한다. qsort는 fcmp가 가리키는 사용자 정의 비교 함수를 사용하여 테이블 내의 요소를 정렬한다.

* base는 정렬될 테이블의 처음(0번째 요소)를 가리킨다.
* nelem은 테이블에 있는 요소의 수를 나타낸디.
* width는 테이블에 있는 각 요소의 크기를 나타낸다.

리 턴 값 없음
이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C 와 도 호환된다.
참 조 `bsearch`,`lsearch`

쌇rand
#함수 RAND

기 능 난수를 발생시킨다.
구 문 int rand(void);
프로토타입 `stdlib.h`
기능 설명 배수합성의 난수발생을 사용해서 0에서 RAND_ MAX 범위 사이에서 연속되는 유사 난수를 발 생 시킨다.
리 턴 값 발생된 유사난수를 리턴한다.
이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C 와 도 호환된다.
참 조 `random`,`radomize`,`srand`

쌇random
#함수 RANDOM

기 능 난수를 발생시킨다.
구 문 int random(int num);
프로토타입 `stdlib.h`
기능 설명 0과 (num-1)간의 난수를 발생시킨다. random (num)은 rand() %(num)으로 정의된 매크로 이 다.
리 턴 값 0과 (num-1) 사이의 난수를 리턴한다.
이 식 성 대응되는 함수가 터보 파스칼에고 있다.
참 조 `rand`,`randomize`,`srand`

쌇randomize
#함수 RANDOMIZE

기 능 난수 발생기를 난수로 초기화한다.
구 문 #include <stdlib.h>
#include <time.h>
void randomize(void);
프로토타입 `stdlib.h`
기능 설명 randomize는 난수 발생기를 난수로 초기화한

다. 이함수는 매크로 이므로 time.h를 포함하여야 한다.

리 턴 값 없음

이 식 성 터보 파스칼 에도 동일한 함수가 있다.

참 조 `srand`,`rand`,`random`

쌇_rotl

#함수 _ROTL

기 능 부호없는 정수를 왼쪽으로 비트 회전시킨다.

구 문 unsigned _rotl(unsigned value,int count);

프로토타입 `stdlib.h`

기능 설명 value로 주어진 값을 count 비트만큼 왼쪽으로 회전시킨다. 회전된 결과는 부호없는 정수이다.

리 턴 값 value의 왼쪽으로 count비트만큼 회전된 결과를 리턴한다.

참 조 `_lrotl`

쌇_rotr

#함수 _ROTR

기 능 부호없는 정수를 오른쪽으로 비트 회전시킨다

구 문 unsigned _rotr(unsigned value,int count);

프로토타입 `stdlib.h`

기능 설명 value로 주어진 값을 count 비트만큼 오른쪽으로 회전시킨다. 회전된 결과는 부호없는 정수 이다.

리 턴 값 value의 오른쪽으로 count비트만큼 회전된 결과를 리턴한다.

참 조 `_lrotr`

쌇srand

#함수 SRAND

기 능 난수 발생기를 초기화한다.

구 문 void srand(unsigned seed);

프로토타입 `stdlib.h`

기능 설명 srand는 인수값 1과 함께 호출함으로써 난수 발생기를 다시 초기화 할수 있다. 또한 seed를 부여함으로써 새로운 난수 발생기의 시작

시점을 설정할 수 있다.
리 턴 값 없음
이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C 와 도 호환된다.
참 조 `rand`,`random`,`randomize`

쌇strtod
#함수 STRTOD

기 능 문자열을 double 값으로 변환한다.
구 문 #include <stdlib.h>
double strtod(const char *s,
char **endptr);
프로토타입 `stdlib.h`
기능 설명 문자열 s를 double값으로 변환한다. 여기서 s 는 double값으로 인식될 수 있는 문자이어야 한다.
리 턴 값 바꿔진 double값을 리턴한다. 오버플로가 생기면 HUGE_VAL을 리턴한다.
이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C 와 도 호환된다.
참 조 `atof`

쌇strtol
#함수 STRTOL

기 능 문자열을 long 값으로 변환한다.
구 문 #include <stdlib.h>
double strtol(const char *s,
char **endptr);
프로토타입 `stdlib.h`
기능 설명 문자열 s를 long값으로 변환한다. 여기서 s는 long 값으로 인식될 수 있는 문자이어야 한다
리 턴 값 바꿔진 long값을 리턴한다. 오버플로가 생기면 HUGE_VAL을 리턴한다.
이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C 와 도 호환된다.
참 조 `atoi`,`atol`,`strtoul`

쌇strtoul
#함수 STRTOUL

기　　능　문자열을 unsigned long 값으로 변환한다.
구　　문　#include <stdlib.h>
　　double strtoul(const char *s,
　　char **endptr);
프로토타입　`stdlib.h`
기능 설명　문자열 s를 unsigned long값으로 변환한다.
　　여기서 s는 unsigned long값으로 인식될 수
　　있는 문자이어야 한다
리 턴 값　바꿔진 unsigned long값을 리턴한다. 오버플
　　로가 생기면 HUGE_VAL을 리턴한다.
이 식 성　UNIX 시스템에서 사용할 수 있으며 ANSI-C와
　　도 호환된다.
참　　조　`atoi`,`atol`,`strtol`

쌇swab
#함수 SWAAB

기　　능　바이트를 교체한다.
구　　문　void swab(char *from,char *to,int nbyte);
프로토타입　`stdlib.h`
기능 설명　from으로부터 to까지 nbyte만큼 복사한다. 이
　　때 서로 인접한 홀짝 바이트를 뒤 바꾼다.
　　이 기능은 바이트 순서가 서로 다른 기종간에
　　자료를 교환할 때 유용하다.
리 턴 값　없음
이 식 성　UNIX 시스템상에서 사용할 수 있다.

쌇ultoa
#함수 ultoa

기　　능　unsigned long을 문자열로 변환한다.
구　　문　char *ultoa(unsigned long value,
　　char *string, int radix);
프로토타입　`stdlib.h`
기능 설명　value를 null 종료 문자열로 바꿔서 그 내용
　　을 string에 저장한다. value는 unsigned lo
　　ng이다.

　　radix는 value를 변환하는데 필요한 밑수이다
　　그 범위는 2에서 32까지 인데 오버플로를 검

사하지 않는다.
리 턴 값 string을 리턴한다. 에러 리턴은 없다.
참 조 `itoa`,`ltoa`

쌇string.h

함수

`memccpy` `memchr` `memcmp`
`memcpy` `memicmp` `memmove`
`memset` `movedata` `movmem`
`setmem` `stpcpy` `strcat`
`strchr` `strcmp` `strcmpi`
`strcpy` `strcspn` `strdup`
`_strerror` `strerror` `stricmp`
`strlen` `strlwr` `strncat`
`strncmp` `strncmpi` `strncpy`
`strnicmp` `strnset` `strpbrk`
`strrchr` `strrev` `strset`
`strspn` `strstr` `strtok`
`strupr`

쌇stpcpy
#함수 STRCPY

기 능 하나의 문자열을 다른 문자열에 복사한다.
구 문 char *strcpy(char *dest, const char *src);
프로토타입 `string.h`
기능 설명 strcpy는 문자열 src를 dest에 복사한다. nul
l 종료 문자를 만날 때 까지 복사를 계속한다
리 턴 값 strcpy는 dest + strlen(src)를 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있다.

쌇strcat
#함수 STRCAT

기 능 하나의 문자열을 다른 문자열 뒤에 붙인다.
구 문 char *strcat(char dest, const char *src);
프로토타입 `string.h`
기능 설명 dest의 끝에 src를 복사하여 붙인다. 그로 인

한 문자열의 길이는 strlen(dest)+strlen(src)와 같다.

리 턴 값 연결된 문자열의 포인터를 리턴한다.

이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C와도 호환된다.

쌇strchr

#함수 STRCHR

기 능 주어진 문자를 찾아 문자열을 탐색한다.

구 문 char *strchr(const *s, int c);

프로토타입 `string.h`

기능 설명 지정된 문자를 찾기위해 문자열을 차례대로 탐색한다. 문자 c를 만날 때까지 문자열 s를 검사한다. 문자열의 끝을 나타내는 null로 문자열의 일부로 간주한다. 예를 들면,

strchr(strs,0);

은 문자열 strs의 끝을 나타내는 null 문자의 포인터를 리턴한다.

리 턴 값 문자열 s에서 문자 c가 최초로 나타나는 위치를 포인터로 리턴한다. 만일 s에 c가 없으면 null을 리턴한다.

이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C와도 호환된다.

쌇strcmp

#함수 STRCMP

기 능 두개의 문자열을 비교한다.

구 문 int strcmp(const char *s1,const char *s2);

프로토타입 `string.h`

기능 설명 문자열 s1과 s2를 부호에 관계없이 비교한다. 비교는 첫무자부터 순서대로 문자끼리 비교하여 서로 다른 문자가 나오거나 문자열에 끝에 도달할 때까지 계속한다.

리 턴 값 다음값을 리턴한다.

< 0 s1이 s2보다 작음

== 0 s1과 s2가 같음

> 0 s1이 s2보다 큼

이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C와 도 호환된다.

쌇strcmpi

#함수 STRCMPI

기 능 두 개의 문자열을 문자체와 관계 없이 비교한다.

구 문 #include <string.h>
int strcmpi(const char *s1,const char *s2);

프로토타입 `string.h`

기능 설명 s1과 s2를 대소문자 구분하지 않고 부호에 관계 없이 비교한다.
(stricmp)와 동일 매크로로 구현됨

리 턴 값 다음값을 리턴한다.

< 0 s1이 s2보다 작음
== 0 s1과 s2가 같음
> 0 s1이 s2보다 큼

이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C와 도 호환된다.

쌇strcpy

#함수 STRCPY

기 능 문자열을 다른곳에 복사한다.

구 문 #include <string.h>
char *strcpy(char *dest, const char *src);

프로토타입 `string.h`

기능 설명 문자열 src를 dest에 복사한다. null종료 문자열을 만나면 복사를 중지한다.

리 턴 값 dest를 리턴한다.

이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C와 도 호환된다.

쌇strcspn

#함수 STRCSPN

기 능 문자열을 검사하여 주어진 문자열의 문자가 전혀 없는 초기 세그먼트를 계산한다.
구 문 #include <string.h>
size_t strcspn(const char *s1, const char *s2);
프로토타입 `string.h`
기능 설명 문자열 s1을 검사하여 주어진 문자열 s2에 들어있는 문자가 전혀 없는 최초의 구간 길이를 구한다.
리 턴 값 문자가 전혀 없는 최초의 구간 길이를 리턴한다.

쌇strdup
#함수 STRDUP

기 능 문자열을 새로 만들어진 영역에 복사한다.
구 문 char *strdup(const char *s);
프로토타입 `string.h`
기능 설명 `malloc`을 호출하여 새로운 공간을 할당받아서 그 장소에 문자열 s의 내용을 저장한다. 할당되는 문자열의 길이는 strlen(s)+1 이다. 사용자는 strdup에 의해 할당된 공간이 더 이상 필요하지 않을경우 그 공간을 해제 시켜 주어야 한다.
리 턴 값 중복된 문자열을 갖는 제공 위치의 포인터를 리턴하거나 공간 할당에 실패했을 경우 null을 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있다.
참 조 `free`

쌇stricmp
#함수 stricmp

기 능 두 개의 문자열을 문자체와 관계 없이 비교한다.
구 문 #include <string.h>
int stricmp(const char *s1,const char *s2);
프로토타입 `string.h`
기능 설명 s1과 s2를 대소문자 구분하지 않고 부호에 관계 없이 비교한다.

리 턴 값 다음값을 리턴한다.

< 0 s1이 s2보다 작음
== 0 s1과 s2가 같음
> 0 s1이 s2보다 큼

이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C와 도 호환된다.

쌇strlen
#함수 STRLEN

기 능 문자열의 길이를 계산한다.
구 문 #include <string.h>
size_t strlen(sonst char *s);
프로토타입 `string.h`
기능 설명 문자열 s의 길이를 리턴한다.
리 턴 값 null종료 문자를 빼고 문자열 s를 이루는 문자의 수를 리턴한다.
이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C와 도 호환된다.

쌇strlwr
#함수 STRLWR

기 능 문자열의 대문자를 소문자로 바꾼다.
구 문 char *strlwr(char *s);
프로토타입 `string.h`
기능 설명 문자열 속의 대문자(A...Z)를 소문자(a...z)로 바꾼다. 이외의 다른 문자는 바꾸지 않는다.
리 턴 값 문자열 s의 포인터를 리턴한다.
참 조 `strupr`

쌇strncat
#함수 STRNCAT

기 능 한 믄자열의 부분을 다른 문자열에 추가한다.
구 문 #include <string.h>
char *strncat(char *dest, const char *src,
size_t maxlen);

프로토타입 `string.h`
기능 설명 src에서 최대 maxlen만큼의 문자를 dest의 뒤에 붙이고, 그 결과 문자열의 최대 길이는 strlen(dest) + maxlen 이 된다.
리 턴 값 dest를 리턴한다.
이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C와도 호환된다.

쌇strncmp
#함수 STRNCMP

기 능 두 개의 문자열을 주어진 길이 만큼만 비교한다.
구 문 #include <string.h>
int strncmp(const char *s1,
const char *s2,size_t maxlen);
프로토타입 `string.h`
기능 설명 strcmp처럼 부호와 관계없이 두 개의 문자열을 비교한다. 단지 maxlen 만큼만 비교한다.
리 턴 값 문자열 s1과 s2의 전체나 일부를 비교하여 그 결과를 기초로 int를 리턴한다.

< 0 s1이 s2보다 작음
== 0 s1과 s2가 같음
> 0 s1이 s2보다 큼

이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C와도 호환된다.

쌇strncmpi
#함수 STRNCMPI

기 능 문자열의 부분을 문자체에 부분 없이 비교한다.
구 문 #include <string.h>
int strncmpi(const char *s1,
const char *s2,size_t n);
프로토타입 `string.h`
기능 설명 부호를 고려하면서 문자열 s1과 s2를 n바이트를 최대 길이로 하여 비교한다. 이 비교는 대소문자를 구분하지 않는다.

리 턴 값 다음값을 리턴한다.

< 0 s1이 s2보다 작음
== 0 s1과 s2가 같음
> 0 s1이 s2보다 큼

쌇strncpy
#함수 STRNCPY

기 능 한 문자열을 지정된 바이트 만큼 다른 문자열로 복사하면서 문자열을 줄이거나 추가시킨다

구 문
```
#include <string.h>
char strncpy(char *dest, char *src,
     size_t maxlen);
```

프로토타입 `string.h`

기능 설명 src로부터 maxlen만큼의 문자를 dest에 복사하면서 dest를 줄이거나 null문자로 채운다. 대상 문자열 dest는 src의 길이가 maxlen 이상이면 null로 끝나지 않을 우려가 있다.

리 턴 값 dest를 리턴한다.

이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C와도 호환된다.

쌇strnicmp
#함수 STRNICMP

기 능 문자열의 부분을 문자체에 부분 없이 비교한다.

구 문
```
#include <string.h>
int strnicmp(const char *s1,
    const char *s2,size_t n);
```

프로토타입 `string.h`

기능 설명 부호를 고려하면서 문자열 s1과 s2를 n바이트를 최대 길이로 하여 비교한다. 이 비교는 대소문자를 구분하지 않는다.

리 턴 값 다음값을 리턴한다.

< 0 s1이 s2보다 작음
== 0 s1과 s2가 같음
> 0 s1이 s2보다 큼

쌇strnset
#함수 STRNSET

기　　능　문자열의 지정된 문자수를 주어진 문자로 설정한다.
구　　문　#include <string.h>
　　char *strnset(char *s,int ch,size_t n);
프로토타입　`string.h`
기능 설명　문자열의 s를 첫번째 바이트부터 n번째 까지 문자 ch로 설정한다. strnset은 n 문자만큼 설정되거나 null 문자를 만나면 중지한다.
리 턴 값　s를 리턴한다.

쌇strpbrk
#함수 STRPBRK

기　　능　한 문자열을 주어진 문자열에 있는 문자가 나올 때까지 검사한다.
구　　문　char *strpbrk(const char *s1,
　　const char *s2);
프로토타입　`string.h`
기능 설명　문자열 s1에서 s2의 문자가 처음으로 나타나는 위치를 검사한다.
리 턴 값　s2의 문자가 처음 발견된 위치의 포인터를 리턴한다. 만일 발견된 문자가 없으면 null이 리턴된다.
이 식 성　UNIX 시스템에서 사용할 수 있으며 ANSI-C와도 호환된다.

쌇strrchr
#함수 STRRCHR

기　　능　문자열에서 주어진 문자가 마지막으로 나타나는 위치를 구한다.
구　　문　char *strrchr(const char *s, int c);
프로토타입　`string.h`
기능 설명　문자열을 뒤에서 부터 문자 c가 처음 나올 때까지 검색한다. strrchr은 문자열 s에 있는 문자 c의 마지막 발생을 찾는다. null 문자도

해당 문자열의 일부로 간주한다.
리 턴 값 문자열 s에 c가 마지막으로 나온 위치의 포인터를 리턴한다.
이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C와도 호환된다.

쌇strrev
#함수 STRREV

기 능 문자열의 앞뒤를 바꾼다.
구 문 char *strrev(char *s);
프로토타입 `string.h`
기능 설명 문자열의 앞뒤를 null문자를 제외하고 앞뒤를 바꾼다.
리 턴 값 역전된 문자열의 포인터를 리턴한다.

쌇strset
#함수 STRSET

기 능 문자열의 모든 문자를 주어진 문자로 설정한다.
구 문 char *strset(char *s,int ch);
프로토타입 `string.h`
기능 설명 null을 제외한 문자열 s의 모든 문자를 ch로 바꾼다.
리 턴 값 s를 리턴한다.
참 조 `setmem`

쌇strspn
#함수 STRSPN

기 능 주어진 문자 집합의 부분 집합인 첫번째 세그먼트를 찾아 문자열을 탐색한다.
구 문 #include <string.h>
size_t strspn(const char *s1,
const char *s2);
프로토타입 `string.h`
기능 설명 문자열 s2에 있는 문자로만 이루어빈 문자열 s1 내의 초기 세그먼트를 탐색한다.
리 턴 값 앞에서 찾아진 초기 세그먼트의 길이를 리턴한다.

이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C와 도 호환된다.

쌇strstr
#함수 STRSTR

기 능 주어진 부분 문자열의 발생을 찾기 위하여 문자열을 탐색한다.
구 문 char *strstr(const char *s1,
const char *s2);
프로토타입 `string.h`
기능 설명 부분 문자열 s2를 앞부분부터 찾아 문자열 s1을 검색한다.
리 턴 값 문자열 s2내의 문자열 s1과 일치하는 부분 문자열의 포인터를 리턴한다. 만일 s2중에 s1과 일치하는 부분이 없으면 null을 리턴한다.
이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C와 도 호환된다.

쌇strtok
#함수 STRTOK

기 능 주어진 문자열에서 주어진 문자로 분리되는 토큰을 첫째 문자열에서 찾는다.
구 문 char *strtok(char *s1,const char *s2);
프로토타입 `string.h`
기능 설명 문자열 s1을 분리자 문자열 s2로부터 주어지는 문자들로 분리하는 0이나 텍스트 토큰으로 구성되어 있다고 가정한다.

strtok는 처음 처음 호출되면 s1의 첫번째 토큰에서 시작하는 문자의 포인터를 리턴하고 s1 안에는 그 토큰의 바로 뒤에 null 문자를 기록한다. 이어서 strtok를 호출하는 경우 인수를 null로 주어 s1내의 모든 토큰을 얻을 수 있다.
리 턴 값 s1의 토큰이 발견될 때마다 포인터를 리턴한다. 만일 더 이상의 토큰이 없으면 null을 리턴한다.
이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C와 도 호환된다.

쌇strupr
#함수 STRUPR

기　　능　문자열의 소문자를 대문자로 변환한다.
구　　문　char *strupr(char *s);
프로토타입　`string.h`
기능 설명　문자열 s내의 소문자(a...z)를 대문자(A...Z)
　　로 변환한다. 다른 문자는 변하지 않는다.
리 턴 값　변환된 s를 리턴한다.
참　　조　`strlwr`

쌇sys\stat.h
#헤더파일 SYS\STAT.H

함수

`fstat`　　`stat`

쌇fstat
#함수 FSTAT

기　　능　개방 파일에 대한 정보를 갖는다.
구　　문　#include <sys\stat.h>
　　int fstat(int handle,
　　struct stat *statbuf);
프로토타입　`sys\stat.h`
기능 설명　handle과 연관된 디렉토리나 개방 파일에 관
　　한 정보를 구조체 stat에 저장한다. setbuf
　　는 sys\stat.h에서 정의된 stat구조체를 가
　　리 키며 그 구조체는 다음과 같은 필드를 포
　　함한다.

　　st_mode　파일 모드에 관한 비트 마스크
　　st_dev　파일이 위치하고 있는 드라이브 번
　　호
　　st_rdev　st_dev와 같다.
　　st_nlink 정수 상수 1이 리턴된다.
　　st_size　바이트 단위의 개방 파일의 크기
　　st_atime 가장 최근에 파일이 수정된 시기
　　st_mtime 상동

st_ctime 상동

리 턴 값 개방 파일에 대한 정보를 성공적으로 검색한 경우 0이 리턴된다. 에러발생시 -1이 리턴되고`errno`는 아래와 같이 설정된다.

EBADF Bad file handle

참 조 `access`,`chmod`,`stat`

쌿stat

#함수 STAT

기 능 열린 파일에 대한 정보를 얻는다.

구 문 #include <sys\stat.h>

int stat(char *path,struct stat *statbuf);

프로토타입 `sys\stat.h`

기능 설명 주어진 파일이나 디렉토리에 관한 정보를 구조체 stat에 저장한다. setbuf는 sys\stat.h에서 정의된 stat구조체를 가리키며 그 구조체는 다음과 같은 필드를 포함한다.

st_mode 파일 모드에 관한 비트 마스크

st_dev 파일이 위치하고 있는 드라이브 번호

st_rdev st_dev와 같다.

st_nlink 정수 상수 1이 리턴된다.

st_size 바이트 단위의 개방 파일의 크기

st_atime 가장 최근에 파일이 수정된 시기

st_mtime 상동

st_ctime 상동

리 턴 값 개방 파일에 대한 정보를 성공적으로 검색한 경우 0이 리턴된다. 에러발생시 -1이 리턴되고`errno`는 아래와 같이 설정된다.

ENOENT 파일 혹은 디렉토리를 찾지 못함

참 조 `access`,`chmod`,`fstat`

쌿sys\timeb.h

함수

`ftime`

쌇ftime
#함수 FTIME

기　　능　현재 시간을 구조체 timeb에 저장한다.
구　　문　#include <sys\timeb.h>
　　void ftime(struct timeb *buf);
프로토타입　`sys\timeb.h`
기능 설명　ftime은 현재의 시간을 결정하고 buf에 의해
　　지정된 구조체 timeb에 있는 필드를 채운다.
　　timeb구조체에는 다음의 필드가 있다.

　　* time　그리니치 표준시로부터 경과된 시간
　　을 초단위로 나타낸다.
　　* millitm　초를 밀리초 단위로 나눈 분수부
　　분이다.
　　* timezone GMT와 지방시와의 시차를 분단위
　　로 나타낸다.
　　* dstflag　일광 절약 시간이 적용이 않되면
　　0으로 설정되고 적용이 되는 경
　　우에는 0이외의 값이 설정된다.
리 턴 값　없음
이 식 성　UNIX 시스템 V에서 사용할 수 있다.
참　　조　`asctime`,`ctime`,`gmtime`,`localtime`,
　　`stime`,`tzset`

쌇time.h
#헤더파일 TIME.H

`asctime`　`clock`　`ctime`
`difftime`　`gmtime`　`localtime`
`stime`　`time`　`tzset`

쌇asctime
#함수 ASCTIME

기　　능　날짜와 시간을 ASCII로 변환한다.
구　　문　#include <time.h>

char *asctime(const struct tm *tblock);

프로토타입 `time.h`

기능 설명 *tblock 구조체에 저장된 시간을 ctime 문자열과 동일한 형태의 문자열로 변환한다.

Sun Sep 16 01:03:52 1973\n\0

리 턴 값 포인터를 날짜와 시간을 포함하는 문자열로 리턴한다. 이 문자열은 asctime으로 각각 호출 되면서 고쳐 쓰여지는 정적 변수이다.

이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C와도 호환된다.

참 조 `ctime`,`difftime`,`ftime`,`gmtime`, `localtime`,`stime`,`time`,`tzset`

쌿clock

#함수 CLOCK

기 능 프로세스 시간을 결정한다.

구 문 #include <time.h>
clock_t clock(void);

프로토타입 `time.h`

기능 설명 두 개의 경우 사이에서 시간 간격을 결정하기 위해 사용된다. 시간을 초 단위로 결정하기 위해서는 clock에 의해 리턴된 값을 매크로 값 CLK_TCK로 나누어 주어야 한다.

리 턴 값 프로그램 호출이 시작된 후 경과된 프로세스 시간을 리턴한다. 프로세스 시간이 사용 불가능 하거나 그 값이 표시될 수 없는 경우에는 -1을 리턴한다.

이 식 성 ANSI-C와 호환된다.

쌿ctime

#함수 CTIME

기 능 일자와 시간을 문자열로 변환시킨다.

구 문 #include <time.h>
char *ctime(const time_t *time);

프로토타입 `time.h`

기능 설명 time에(time 함수 리턴값) 의해 지정된 시간을 26자의 문자열로 다음과 같이 변환한다.

이 때 새로운 행의 문자와 null 문자로 종료
한다.

Mon Nov 21 11:11:54 1983\n\0

리 턴 값 일자와 시간을 갖고있는 문자열의 포인터를 리턴한다. 이 리턴값은 ctime 호출시마다 중복되어 써지는 정적데이터의 일부이다.

이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C와도 호환된다.

참 조 `asctime`,`difftime`,`ftime`,`getdate`, `gmtime`,`localtime`,`settime`,`time`, `tzset`

쌿difftime

#함수 DIFFTIME

기 능 두 시간대간의 시차를 계산한다.

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

프로토타입 `time.h`

기능 설명 time1부터 time2까지 경과한 시간을 초단위로 계산한다.

리 턴 값 계산결과를 double형으로 리턴한다.

이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C와도 호환된다.

참 조 `asctime`,`ctime`,`time`

쌿gmtime

#함수 GMTIME

기 능 그리니치 표준치(GMT)로 일자와 시간을 변환한다.

구 문 #include <time.h>
struct tm *gmtime(const time_t *timer);

프로토타입 `time.h`

기능 설명 time에 의해 리턴된 값이 어드레스를 받아서 분할된 시간을 갖고 있는 int형 구조체의 포인터를 리턴한다.

리 턴 값 분할된 시간을 포함하는 구조체의 포인터를 리턴한다. 이 구조체는 매번 호출시마다 변경

되는 정적인 구조체이다.
이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C와 도 호환된다.
참 조 `asctime`,`ctime`,`ftime`,`localtime`, `stime`,`time`,`tzset`

쌇localtime
#함수 LOCALTIME

기 능 일자와 시간을 구조체로 변환한다.
구 문 #include <time.h>
struct tm *localtime(const time_t *timer);
프로토타입 `time.h`
기능 설명 time에 의해 리턴되는 값이 어드레스를 받아서 시간을 포함하고 있는 int형 구조체에 대한 포인터를 리턴한다. 이 함수는 지역시간과 일광 절약 시간을 조정한다.

리 턴 값 분할된 시간을 포함하는 구조체에 대한 포인터를 리턴한다. 이 구조체는 매번 호출 때마다 내용이 변경되는 정적인 구조체 이다.
이 식 성 UNIX 시스템에서 사용할 수 있으며 ANSI-C와 도 호환된다.
참 조 `asctime`,`ctime`,`ftime`,`gmtime`,`stime`, `time`,`tzset`

쌇stime
#함수 STIME

기 능 시스템의 날자와 시간을 정한다.
구 문 #include <time.h>
int stime(time_t *tp);
프로토타입 `time.h`
기능 설명 stime은 시스템의 날짜와 시간을 정한다. tp는 GMT로 1970년 1월 1일 00:00:00부터 초단위 까지 측정된 시간을 가리킨다.
리 턴 값 stime은 0을 리턴한다.
이 식 성 UNIX 시스템상에서 사용할 수 있다.
참 조 `asctime`,`ftime`,`gettime`,`gmtime`, `localtime`,`time`,`tzset`

쌇time
#함수 TIME

기　　능　경과한 시간을 구한다.
구　　문　#include <time.h>
　　time_t time(time_t *timer);
프로토타입　`time.h`
기능 설명　GMT로 1970년 1월 1일 이후의 현재 시간을 초 단위까지 알려준다. 만일 timer가 null이 아니면 그 값이 가리키는 위치에 시간을 정한다
리 턴 값　시간을 리턴한다.
이 식 성　UNIX 시스템에서 사용할 수 있으며 ANSI-C와도 호환된다.
참　　조　`asctime`,`ctime`,`difftime`,`ftime`, `gettime`,`gmtime`,`localtime`,`settime`, `stime`,`tzset`

쌇tzset
#함수 TZSET

기　　능　전역변수 daylight, timezone, tzname에 값을 설정한다.
구　　문　#include <time.h>
　　void tzset(void);
프로토타입　`time.h`
기능 설명　환경변수 TZ에 의거하여 전역 변수 daylight, timezone, tzname의 값을 결정한다.
리 턴 값　없음
이 식 성　UNIX와 XENIX시스템에서 사용할 수 있다.
참　　조　`asctime`,`ctime`,`ftime`,`gmtime`, `localtime`,`stime`,`time`

쌇values.h
#헤더파일 VALUES.H

VALUES.H 에는 UNIX와 XENIX에서의 호환을 위하여 float에서 double 까지의 상수가 정의되어 있다.

BITSPERBYTE　　FMAXEXP
_DEXPLEN　　FMAXPOWTWO
DMAXEXP　　FMINEXP

DMAXPOWTWO FSIGNIF
DMINEXP HIBITI
DSIGNIF HIBITL
_EXPBASE HIBITS
_FEXPLEN MAXDOUBLE
MAXFLOAT MAXSHRT
MAXINT MINDOUBLE
MAXLONG MINFLOAT

쌿전역변수

`_argc` `_argv` `daylight`
`directvideo` `_8087` `environ`
`errno` `_doserror` `_fmode`
`_heaplen` `_osmajor` `_osminor`
`_psp` `_stklen` `timezone`
`tzname` `_version`

쌿_fmode
#전역변수 _FMODE

기 능 디폴트 파일 변환 모드를 결정한다.
구 문 extern int _fmode;
선언 파일 `fcntl.h`
기능 설명 텍스트 또는 이진 모드로 파일의 열림과 변환
을 결정한다. _fmode의 디폴트 값은 O_TEXT가
되며 이것은 파일이 텍스트 파일로 읽혀지도
록 한다.

쌿함수

알파벳순 터보-C 함수

`abort` `abs` `absread`
`abswrite` `access` `allocmem`
`arc` `asctime` `asin`
`assert` `atan` `atan2`
`atexit` `atof` `atoi`
`atol` `bar` `bar3d`
`bdos` `bdosptr` `bioscom`
`biosdisk` `biosequip` `bioskey`

`biosmemory` `biosprint` `biostime`
`brk` `bsearch` `cabs`
`calloc` `ceil` `cgets`
`chdir` `_chmod` `chmod`
`chsize` `circle` `_clear87`
`cleardevice` `clearerr` `clearviewport`
`clock` `_close` `closegraph`
`clreol` `clrscr` `_constrol87`
`coreleft` `cos` `cosh`
`country` `cprintf` `cputs`
`_creat` `creat` `creatnew`
`creattemp` `cscanf` `ctime`
`ctrlbrk` `delay` `detectgraph`
`difftime` `disable` `div`
`dosexterr` `dostounix` `drawpoly`
`dup` `dup2` `ecvt`
`ellipse` `__emit__` `enable`
`eof` `exec` `_exit`
`exit` `exp` `fabs`
`farcalloc` `farcoreleft` `farfree`
`farmalloc` `farrealloc` `fclose`
`fcloseall` `fcvt` `fdopen`
`feof` `ferror` `fflush`
`fgetc` `fgetchar` `fgetpos`
`fgets` `filelength` `fileno`
`fillellipse` `fillpoly` `findfirst`
`findnext` `floddfill` `floor`
`flushall` `fmod` `fnmerge`
`fnsplit` `fopen` `FP_OFF`
`_fpreset` `fprintf` `FP_SEG`
`fputc` `fputchar` `fputs`
`fread` `free` `freemem`
`freopen` `frexp` `fscanf`
`fseek` `fsetpos` `fstat`
`ftell` `ftime` `fwrite`
`gcvt` `geninterrupt` `getarccoords`
`getaspectratio` `getbkcolor` `getc`
`getcbrk` `getch` `getchar`
`getche` `getcolor` `getcurdir`
`getcwd` `getdate` `getdefaultpalette`
`getdfree` `getdisk` `getdrivername`

`getdta` `getenv` `getfat`
`getfatd` `getfillpattern` `getfillsettings`
`getftime` `getgraphmode` `getimage`
`getlinesettings` `getmaxcolor` `getmaxmode`
`getmaxx` `getmaxy` `getmodename`
`getmoderange` `getpalette` `getpalettesize`
`getpass` `getpixel` `getpsp`
`gets` `gettext` `gettextinfo`
`gettextsettings` `gettime` `getvect`
`getverify` `getviewsettings` `getw`
`getx` `gety` `gmtime`
`gotoxy` `graphdefaults` `grapherrmsg`
`_graphfreemem` `_graphgetmem` `graphresult`
`harderr` `hardresume` `hardretn`
`highvideo` `hypot` `imagesize`
`initgraph` `inport` `inportb`
`insline` `installuserdrive`
`installuserfont` `int86` `int86x`
`intdos` `intdosx` `intr`
`ioctl` `isalnum` `isalpha`
`isascii` `isatty` `iscntrl`
`isdigit` `isgraph` `islower`
`isprint` `ispunct` `isspace`
`isupper` `isxdigit` `itoa`
`kbhit` `keep` `labs`
`ldiv` `lfind` `line`
`linerel` `lineto` `localtime`
`lock` `log` `log10`
`longjmp` `lowvideo` `_lrotl`
`_lrotr` `lsearch` `lseek`
`ltoa` `malloc` `_matherr`
`max` `memccpy` `memchr`
`memcmp` `memcpy` `memicmp`
`memmove` `memset` `min`
`mkdir` `MK_FP` `mktemp`
`modf` `movedata` `moverel`
`movetext` `moveto` `movmem`
`normvideo` `nosound` `_open`
`open` `outport` `outportb`
`outtext` `outtextxy` `parsfnm`
`peek` `peekb` `perror`

`pieslice` `poke` `pokeb`
`poly` `pow` `pow10`
`printf` `putc` `putch`
`putchar` `putenv` `putimage`
`putpixel` `puts` `puttext`
`putw` `qsort` `raise`
`rand` `randbrd` `randbwr`
`random` `radomize` `_read`
`read` `realloc` `rectangle`
`registerbgidriver``registerbgifont` `remove`
`rename` `restorecrtmode` `rewind`
`rmdir` `_rotl` `_rotr`
`sbrk` `scanf` `searchpath`
`sector` `segread` `setactivepage`
`setallpalette` `setaspectratio` `setbkcolor`
`setblock` `setbuf` `setvbrk`
`setcolor` `setdate` `setdisk`
`setdta` `setfillpattern` `setfillstyle`
`setftime` `setgraphbufsize` `setgraphmode`
`setjmp` `setlinestyle` `setmem`
`setmode` `setpalette` `setrgbpalette`
`settextjustify` `settextstyle` `settime`
`setusercharsize` `setvbuf` `setvect`
`setverify` `setviewport` `setvisualpage`
`setwritemode` `signal` `sin`
`sinh` `sleep` `sopen`
`sound` `spawn` `sprintf`
`sqrt` `srand` `sscanf`
`stat` `_status87` `stime`
`strcpy` `strcat` `strchr`
`strcmp` `strcmpi` `strcpy`
`strcspn` `strdup` `_strerror`
`strerror` `stricmp` `strlen`
`strlwr` `strncat` `strncmp`
`strncmpi` `strncpy` `strnicmp`
`strnset` `strnchr` `strrev`
`strset` `strspn` `strstr`
`strtod` `strtok` `strtol`
`strtoul` `strupr` `swab`
`system` `tan` `tanh`
`tell` `textattr` `textbackground`

`textcolor` `texthight` `textmode`
`textwidth` `time` `tmpfile`
`tmpnam` `toascii` `_tolower`
`tolower` `_toupper` `toupper`
`tzset` `ultoa` `ungetc`
`ungetch` `unixtodos` `unlink`
`unlock` `va_` `vfprintf`
`vfscanf` `vsprintf` `vsscanf`
`wherex` `wherey` `window`
`_write`

쌿기능별함수
#기능별 함수 분류

`분류 루틴`(Classification routines)
`변환 루틴`(Conversion routines)
`디렉토리 루틴`(Directory control routines)
`진단 루틴`(Diagnostic routines)
`그래픽 루틴`(Graphics routines)
`입출력 루틴`(Input/Output routines)
`인터페이스 루틴`(Interface routines)
`메모리 처리 루틴`(manipulation routine)
`수학 루틴`(Math routines)
`메모리 할당 루틴`(Memory allocation routines)
`프로세스 제어 루틴`(Process control routines)
`표준 루틴`(Standard routines)
`텍스트 윈도 디스플레이 루틴`
(Text window display routines)
`시간과 날짜 루틴`(Time and date routines)
`가변 인수 리스트 루틴`
(Variable argument list routines)
`기타 루틴`(etc)

쌿분류 루틴
#분류 루틴

`isalnum` `isalpha` `isascii`
`iscntrl` `isdigit` `isgraph`
`islower` `isprint` `ispunct`
`isspace` `isupper` `isxdigit`

쌿변환 루틴
#변환 루틴

`atof` `atoi` `atol`
`ecvt` `fcvt` `gcvt`
`itoa` `ltoa` `strtod`
`strtol` `strtoul` `_tolower`
`toascii` `tolower` `_toupper`
`toupper` `ultoa`

쌿디렉토리 루틴
#디렉토리 루틴

`chdir` `findfirst` `findnext`
`fnmerge` `fnsplit` `getcurdir`
`getcwd` `getdisk` `mkdir`
`mktemp` `rmdir` `searpath`
`setdisk`

쌿진단 루틴
#진단 루틴

`assert` `matherr` `perror`

쌿그래픽 루틴
#그래픽 루틴

`arc` `bar` `bar3d`
`circle` `cleardevice` `clearviewport`
`closegraph` `detectgraph` `drawpoly`
`ellipse` `fillellipse` `fillpoly`
`floodfill` `getarccoords` `getbkcolor`
`getcolor` `getdefaultpalette` `getdrivername`
`getfillpattern` `getfillsettings` `getgraphmode`
`getimage` `getlinesettings` `getmaxcolor`
`getmaxmode` `getmaxx` `getmaxy`
`getmodename` `getmoderange` `getpalette`
`getpalettesize` `getpixel` `gettextsettings`
`getviewsettings` `getx` `gety`
`graphdefaults` `grapherrormsg` `_graphfreemem`
`_graphgetmem` `graphresult` `imagesize`

`initgraph` `installuserdriver` `installuserfont`
`line` `linerel` `lineto`
`moverel` `moveto` `outtext`
`outtextxy` `pieslice` `putimage`
`rectangle` `registerbgidriver` `registerbgifont`
`restorecrtmode` `sector` `setactivepage`
`setallpalette` `setaspectratio` `setbkcolor`
`setcolor` `setfillpattern` `setfillstyle`
`setgraphbufsize` `setgraphmode` `setlinestyle`
`setpalette` `setrgbpalette` `settextjustify`
`settextstyle` `setusercharsize` `setviewport`
`setvisualpage` `setwritemode` `texthight`
`textwidth`

쌿입출력 루틴
#입출력 루틴

`access` `cgets` `_chmod`
`chmod` `chsize` `clearerr`
`close` `_close` `cprintf`
`cputs` `creat` `_creat`
`creatnew` `creattemp` `cscanf`
`dup` `dup2` `eof`
`fclose` `fcloseall` `fdopen`
`feof` `ferror` `fflush`
`fgetc` `filelength` `fileno`
`fluahall` `fopen` `fprintf`
`fputc` `fputchar` `fread`
`freopen` `fscanf` `fseek`
`fsetpos` `fstat` `ftell`
`fwrite` `getc` `getch`
`getchar` `getche` `getftime`
`getpass` `gets` `getw`
`ioctl` `isatty` `kbhit`
`lock` `lseek` `open`
`_open` `perror` `printf`
`putc` `putch` `putchar`
`puts` `putw` `_read`
`read` `remove` `rename`
`rewind` `scanf` `setftime`
`setmode` `setvbuf` `sopen`

`sprintf` `sscanf` `stat`
`_strerror` `strerror` `tell`
`tmpfile` `tmpnam` `ungetc`
`ungetch` `unlock` `vfprintf`
`vfscanf` `vprintf` `vscanf`
`vsprintf` `vsscanf` `_write`
`write`

쨟인터페이스 루틴
#인터페이스 루틴

`absread` `abswrite` `bdos`
`bdosptr` `bioscom` `biosdisk`
`biosequip` `bioskey` `biosmemory`
`biosprint` `biostime` `country`
`ctrlbrk` `disable` `dosexterror`
`enable` `FP_OFF` `FP_SEG`
`freemem` `geninterrupt` `getcbrk`
`getdfree` `getdta` `getfat`
`getfatd` `getpsp` `getvect`
`getverify` `harderr` `hardresume`
`hardretn` `inport` `inportb`
`int86` `int86x` `intdos`
`intdosx` `intr` `keep`
`MK_FP` `outport` `outportb`
`parsfnm` `peek` `peekb`
`poke` `pokeb` `randbrd`
`randbwr` `segread` `setcbrk`
`setdta` `setvect` `setverify`
`sleep` `unlink`

쨟메모리 처리 루틴
#메모리 처리 루틴

`memccpy` `memchr` `memcmp`
`memicmp` `memmove` `setmem`
`strcpy` `strcat` `strchr`
`strcmp` `stricmp` `strcpy`
`strcspn` `strdup` `stricmp`
`strlen` `strlwr` `strncat`
`strncmp` `strncmpi` `strncpy`

`strnicmp` `strnset` `strpbrk`
`strnchr` `strrev` `strset`
`strspn` `strstr` `strtok`
`strupr`

쌿수학 루틴
#수학 루틴

`abs` `acos` `asin`
`atan` `atan2` `atof`
`atol` `cabs` `ceil`
`_clear87` `_control87` `cos`
`cosh` `div` `exp`
`fabs` `fcvt` `floor`
`fmod` `fpreset` `fmod`
`frexp` `gcvt` `hypot`
`labs` `ldiv` `log`
`log10` `_lrotl` `_lrotr`
`ltoa` `_matherr` `matherr`
`modf` `poly` `pow`
`pow10` `rand` `random`
`randomize` `_rotl` `sin`
`sinh` `sqrt` `srand`
`_status87` `strtod` `strtol`
`strtoul` `tan` `tanh`
`ultoa`

쌿메모리 할당 루틴
#메모리 할당 루틴

`allocmem` `brk` `calloc`
`coreleft` `farcalloc` `farcoreleft`
`farfree` `farmalloc` `farrealloc`
`free` `malloc` `realloc`
`sbrk` `setblock`

쌿프로세스 제어 루틴
#프로세스 제어 루틴

`abort` `exec` `_exit`
`exit` `raise` `signal`

`spawn` `system`

쌇표준 루틴
#표준 루틴

`abort` `abs` `atexit`
`atof` `atoi` `atol`
`bsearch` `calloc` `acvt`
`_exit` `exit` `fcvt`
`free` `gcvt` `getenv`
`itoa` `labs` `lfind`
`lsearch` `ltoa` `malloc`
`putenv` `qsort` `rand`
`realloc` `srand` `strtod`
`strtol` `swab` `system`
`ultoa`

쌇텍스트 윈도 디스플레이 루틴
#텍스트 윈도 디스플레이 루틴

`clreol` `clrscr` `deline`
`gettext` `gettextinfo` `getoxy`
`highvideo` `insline` `lowvideo`
`movetext` `normvideo` `puttext`
`textattr` `textbackground` `textcolor`
`textmode` `wherex` `wherey`
`window`

쌇시간과 날짜 루틴
#시간과 날짜 루틴

`asctime` `ctime` `difftime`
`dostounix` `ftime` `getdate`
`gettime` `gmtime` `localtime`
`setdate` `settime` `stime`
`time` `tzset` `unixtodos`

쌇가변 인수 리스트 루틴
#가변 인수 리스트 루틴

`va_arg` `va_end` `va_start`

쌇기타 루틴
#기타 루틴

`delay` `longjmp` `nosound`
`setjmp` `sound`
쌇명령어라인 옵션
#명령어라인 옵션

터보 C 컴파일러의 옵션에는 몇개의 범주가 있으며 대개 다
음과 같이 나눌수 있다.

*`메모리 모델`
*`매크로 정의`
*`최적화 옵션`
*`코드 생성 옵션`
*`소스 코드 옵션`
*`에러 옵션`
*`세그먼트 제어 옵션`

터보 C의 모든 옵션을 보기 위해서는 도스상에서 TCC <ente
r>를 치면된다.

```
------------------------------------------------------
Turbo C  Version 2.0  Copyright (c) 1987, 1988
                      Borland International
Syntax is: TCC [ options ] file[s]
           * = default; -x- = turn switch x off

        -1      80186/286 Instructions
        -A      Disable non-ANSI extensions
        -B      Compile via assembly
        -C      Allow nested comments
        -Dxxx   Define macro
        -Exxx   Alternate assembler name
        -G      Generate for speed
        -Ixxx   Include files directory
        -K      Default char is unsigned
        -Lxxx   Libraries directory
        -M      Generate link map
        -N      Check stack overflow
        -O      Optimize jumps
```

```
        -S      Produce assembly output
        -Uxxx   Undefine macro
        -Z      Optimize register usage
        -a      Generate word alignment
        -c      Compile only
        -d      Merge duplicate strings
        -exxx   Executable file name
        -f    * Floating point emulator
        -f87    8087 floating point
        -gN     Stop after N warnings
        -iN     Maximum identifier length N
        -jN     Stop after N errors
        -k      Standard stack frame
        -lx     Pass option x to linker
        -mc     Compact Model
        -mh     Huge Model
        -ml     Large Model
        -mm     Medium Model
        -ms   * Small Model
        -mt     Tiny Model
        -nxxx   Output file directory
        -oxxx   Object file name
        -p      Pascal calls
        -r    * Register variables
        -u    * Underscores on externs
        -v      Source level debugging
        -w      Enable all warnings
        -wxxx   Enable warning xxx
        -w-xxx  Disable warning xxx
        -y      Produce line number info
        -zxxx   Set segment names
```

쌿메모리 모델

#메모리 모델

이것은 사용자로 하여금 어떤 메모리 모델하에서 TURBO-C가 프로그램을 컴파일 할 것인가를 지정하게 해 준다. 그 모델은 tiny, small, medium, compact, large, huge 가된다.

 ◆-mc

 compact 메모리 모델을 사용하여 컴파일 한다.

 -mh
 huge 메모리 모델을 사용하여 컴파일 한다.

 -ml
 large 메모리 모델을 사용하여 컴파일 한다.

 -mm
 medium 메모리 모델을 사용하여 컴파일 한다.

 -ms
 small 메모리 모델을 사용하여 컴파일 한다.

 -mt
 tiny 메모리 모델을 사용하여 컴파일 한다. small 메모리
 모델과 같은 코드를 생성하지만 tiny 모델 프로그램을 생
 성하기 위해서 링크 수행시에 COT.OBJ를 사용한다.

쌿매크로 정의
#매크로 정의

 -Dxxx
 지정된 식별자 xxx를 단일 공백 문자로 구성되는 문자열로
 정의한다.

 -Dxxx=string
 지정된 식별자 xxx를 등호 기호(=) 뒤에 문자열인 string
 으로 정의한다. string은 공백이나 탭은 포함하지 않는다.

 -Uxxx
 이전에 정의된 식별자의 이름인 xxx를 취소한다.

쌿코드 생성 옵션
#코드 생성 옵션

 -I
 Turbo-C가 확장된 80286 명령을 생성하도록 한다. 이 명령
 은 DOS하에서 IBM-PC/AT처럼 실제 모드로 수행되는 80286
 프로그램을 생성하는 데 사용된다.

 -a

데이터의 할당을 워드 경계로 한다. 디폴트는 off 이고 bi
twise 정렬을 허용한다.

-d
하나의 문자열과 동일한 문자열이 있는 경우 합병한다. 디
폴트는 off 이다. 이 기능을 사용하면 프로그램의 크기를
줄일 수 있다.

-f87
인라인 8087/80287 명령을 생성한다. 부동 소숫점 연산을
하기위해 8087 에뮬레이션 라이브러리를 호출하는 대신에
인라인 8087 명령을 사용한다.

-f
실행 시스템이 8087 시스템을 내장하고 있지 않은 경우에
부동 소숫점 연산을 에뮬레이트 한다.

-f-
부동소숫점 연산을 하지 않으며 따라서 부동 소숫점 라이
브러리를 연결하지 않는다.

-K
모든 char 선언을 unsigned char 형으로 취급한다.

-k
표준 스택 프레임(standard stack frame)을 생성한다. 이
것은 호출된 서브루틴의 스택을 통해서 역으로 추적 하기
위한 디버거 사용시 유용하다.

-N
각 함수 입력에 오버플로 로직을 생성한다. 스택 오버플로
가 발생했을 때 이것을 알려주는 역활을 한다.

-p
함수의 호출과 종료 코드를 Pascal 형식으로 한다. 이럴때
의 장점은 빠르고 작은 코드를 만들어 낼 수 있으나 인수
의 정확한 수와 형을 전달해야 한다. 다시 말해 융통성이
줄어든다.

-u
선언된 함수명 앞에 밑줄(_)을 붙여서 목적 파일에 수록한

다. 초보자는 이 옵션을 사용하지 않는것이 좋다.

 -y
 기호 처리 디버거용의 목적 파일에 행 번호를 포함한다.
 이것은 목적 파일의 크기를 증가 시키지만 실행 프로그램
 의 크기나 속도에는 영향을 주지 않는다.

 -v
 Turbo-C 통합 디버거용의 디버그 정보를 목적 파일에 출력
 한다.

쌿소스 코드 옵션
#소스 코드 옵션

 -A
 ANSI 호환성의 코드를 컴파일한다.

 -C
 주석문의 네스트를 허용한다. 보통 주석문의 네스트는 허
 용 되지 않을 수도 있다.

쌿에러 옵션
#에러 옵션

 -g#
 #에 해당하는 수만큼의 경고와 에러가 발생하면 컴파일을
 중지한다.

 -j#
 #에 해당하는 수만큼의 에러가 발생하면 컴파일을 중지한
 다.

쌿최적화 옵션
#최적화 옵션

 -G
 크기보다는 속도에 중점을 두어 최적화한다. 실행 속도를
 우선으로 한 최적화를 한다.

 -O
 불 필요한 점프를 삭제하거나 스위치문을 재편성함으로써

최적화를 행한다.

-r-
레지스터 변수의 사용을 금지한다.

-Z
레지스터의 내용을 기억하고 가능한 한 그들을 다시 사용
함으로써 불필요한 작동부하를 줄인다.
이 옵션을 사용할 때 사용자는 주의를 요한다. 왜냐하면
레지스터가 포인터에 의해 간접적으로 무효화 되는 경우
컴파일러는 이를 탐지할수 없기 때문이다.

쌿세그먼트 제어 옵션
#세그먼트 제어 옵션

-zAname
코드 세그먼트의 분류(class)명을 name으로 변경시킨다.
디폴트로 코드 세그먼트는 CODE로 분류 지정된다.

-zBname
초기화되지 않은 데이터 세그먼트의 분류명을 name으로 변
경시킨다. 디폴트로 초기화 되지 않은 데이타 세그먼트는
BBS로 분류 지정된다.

-zCname
코드 세그먼트명을 name으로 변경시킨다. 디폴트로 코드
세그먼트 명은 TEXT가 되고 medium,large 그리고 huge 모
델에서는 filename_TEXT 이다. filename은 소스 파일명을
말한다.

-zDname
초기화되지 않은 데이터 세그먼트의 그룹명을 name으로 변
경시킨다. 디폴트로 초기화 되지 않은 데이타 세그먼트의
그룹명은 DGROUP이된다. huge모델 에서는 데이터 그룹이
없다.

-zPname
출력 파일을 name으로 지정된 코드 세그먼트를 위해 코드
그룹으로 생성되게 한다.

-zRname

초기화된 데이타 세그먼트명을 name으로 설정한다. 디폴트로 초기화된 데이터 세그먼트명은 _DATA가 된다. huge모델에서는 세그먼트명이 filename_DATA가 된다.

쌇