리눅스
오류 처리하기

2007. 11. 28

안 효 창

라이브러리 함수의 오류 번호 얻기

errno 변수
기능
오류 번호를 저장한다.
기본형
extern int errno;
헤더 파일
<errno.h>

라이브러리 함수	호출에 실패 했을 때	함수 예
정수 값을 반환하는 함수	-1 반환	open 함수
포인터를 반환하는 함수	NULL 반환	fopen 함수

라이브러리 함수의 오류 번호 얻기

- 19-1

```
#include <stdio.h>
#include <errno.h>   /* errno 변수를 선언하는 헤더
      파일 */

main()
{
   FILE *fp;

   /* fopen 함수 호출에 실패하면 NULL을 반환 */
   if ((fp=fopen("nodata", "r")) == NULL) {
      printf("errno = %d₩n", errno);  /* 오류발생하면
       errno에 오류 번호 저장 */      exit(1);   /* 종료
       하기 */
   }
   exit(0);   /* 종료하기 */
}
```

- 19-2

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <errno.h>

main()
{
   int fd;
   /* nodata 파일을 읽기 전용으로 연다. */
   if ((fd=open("nodata", O_RDONLY)) == 1){ /*
       open 함수 호출에 실패하면 -1 반환 */
      printf("errno = %d₩n", errno);   /* 오류 발생하
       면 errno에 오류 번호 저장 */
      exit(1);
   }
   exit(0);
}
```

라이브러리 함수의 오류 번호 얻기

- 19-3

```
#include <stdio.h>
#include <math.h>   /* sqrt 함수를 정의하는 헤더 파일 */
#include <errno.h>   /* errno 변수를 선언하는 헤더 파일 */

main()
{
   double y;

   errno = 0;  /* errno를 0으로 초기화 */
   y = sqrt(-1);  /* sqrt 함수의 인수로 음수 값이 올 수 없으므로 호출 오류 발생 */

   /* errno가 0이 아니라는 것은 오류가 발생했음을 의미 */
   if (errno != 0) {
      printf("errno = %d₩n", errno);  /* errno에 저장된 오류 번호 출력 */
      exit(1);
   }
   exit(0);
}
```

라이브러리 함수의 오류 번호 얻기

- errno.h

```
#define EPERM 1 /* Operation not permitted */
#define ENOENT 2 /* No such file or directory */
#define ESRCH 3 /* No such process */
#define EINTR 4 /* Interrupted system call */
#define EIO 5 /* I/O error */
#define ENXIO 6 /* No such device or address */
#define E2BIG 7 /* Arg list too long */
:
#define ENOMEDIUM 123 /* No medium found */
#define EMEDIUMTYPE 124 /* Wrong medium type */
```

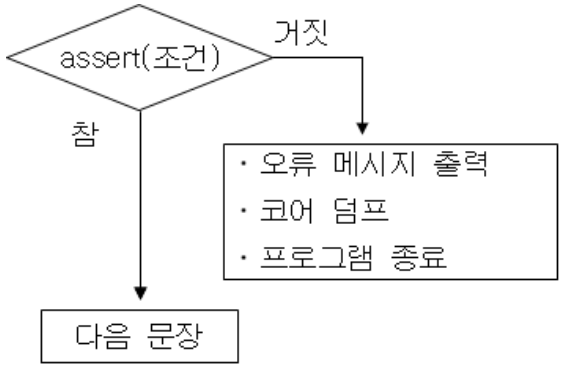
라이브러리 함수의 오류 번호 얻기

- 19-4

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <errno.h>

main()
{
   int fd;

   /* open 함수 호출에 실패하면 -1이 반환 */
   if ((fd=open("nodata", O_RDONLY)) == -1) {
      /* ENOENT는 '그와 같은 파일 또는 디렉토리가 없다'는 의미의 매크로 */
      if (errno == ENOENT)
         printf("nodata is not exist₩n");
      else
         printf("unexpected error: errno = %d₩n", errno);
      exit(1);
   }
   exit(0);
}
```

오류 메시지 출력하고 종료하기

- assert 함수
  기능
    expression이 거짓이면 오류 메시지 출력, 코어 덤프하고 프로그램 종료한다.
  기본형
    void assert(int expression);
    expression : 참과 거짓을 판별하는 수식
  반환값
    없음
  헤더 파일
    <assert.h>



오류 메시지 출력하고 종료하기

- 19-5

```
#include <stdio.h>
#include <assert.h>   /* assert 함수를 정의하는 헤
      더 파일 */
main()
{
   FILE *fp;
   fp=fopen("yesdata", "r");
   /* fp가 참이므로 아무 일 하지 않고 다음 문장을 실
      행한다. */
   assert(fp);
   printf("yesdata exist₩n");
   fclose(fp);   /* fp 닫음 */
   fp=fopen("nodata", "r");  /* nodata 없으므로
      NULL 반환 */
   assert(fp);  /* fp 가 거짓이므로 오류 메시지 출력,
      코어 덤프하고 종료 */
   /* 이후 문장 실행되지 않음 */
   printf("nodata exist₩n");
   fclose(fp);
   exit(0);
}
```

- 19-6

```
#include <stdio.h>
#define NDEBUG  /* NDEBUG를 정의하면 assert
      호출 문장을 실행하지 않음 */
#include <assert.h>

main()
{
   FILE *fp;

   fp=fopen("yesdata", "r");
   assert(fp);  /* 실행되지 않음 */
   printf("yesdata exist₩n");
   fclose(fp);

   fp=fopen("nodata", "r");
   assert(fp);  /* 실행되지 않음 */
   printf("nodata exist₩n");
   fclose(fp);

   exit(0);
}
```

무조건 종료하기

abort 함수

기능

코어 덤프하고 프로그램 종료한다.

기본형

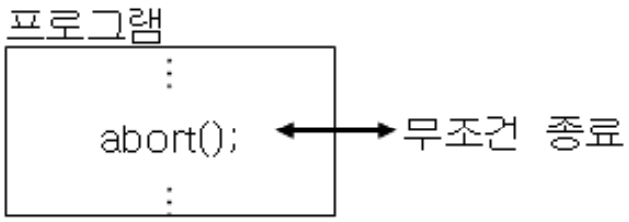
void abort(void);

반환값

없음

헤더 파일

<stdlib.h>



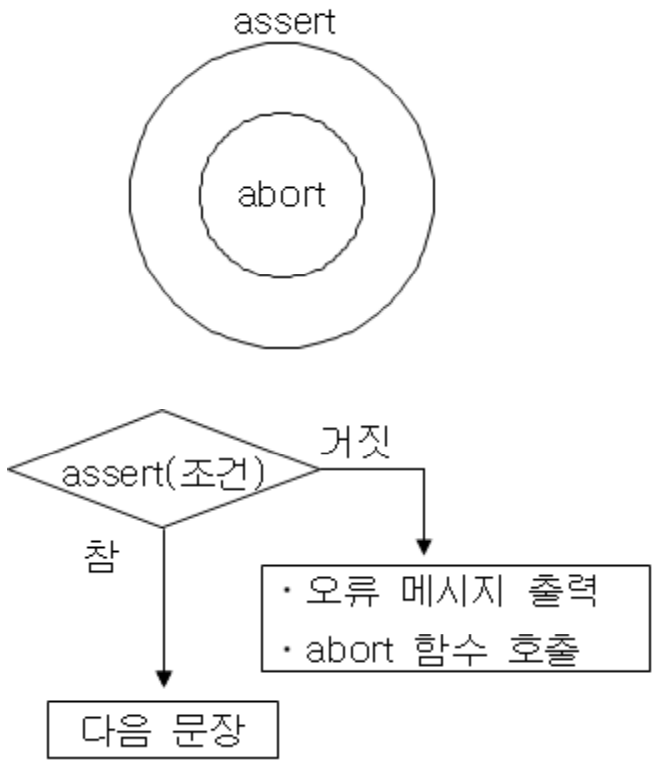
무조건 종료하기

- 19-7

```
#include <stdio.h>
#include <stdlib.h>   /* abort 함수를 정의하는 헤더 파일 */

main()
{
   abort();  /* 코어 덤프하고 프로그램 종료 */
   /* 이후 문장 실행되지 않음 */
   printf("not run₩n");
}
```

무조건 종료하기

- assert 함수에서 abort 함수 이용



무조건 종료하기

- 19-8
  - __FILE__은 소스 파일의 이름을, __LINE__는 현재 라인 번호를 의미하는 매크로

```
#include <stdio.h>
#include <stdlib.h>   /* abort 함수를 정의하는 헤더 파일 */
void my_assert(int expression);

main()
{
   my_assert(7 == 7);
   printf("yes₩n");

   my_assert(3 == 5);
   printf("no₩n");
}

void my_assert(int expression)
{
   if(!(expression)) {  /* expression이 거짓이면 */
      /* __FILE__은 소스 파일, __LINE__은 현재 라인 */
      printf("%s:%d:Assertion failed.₩n", __FILE__, __LINE__);
      abort();
   }
}
```

오류 번호로 오류 원인 얻기

- strerror 함수
  기능
    오류 번호를 설명하는 문자열을 반환한다.
  기본형
    char *strerror(int errnum);
    errnum : 오류 번호
  반환값
    성공 : 오류 번호를 설명하는 문자열
    실패 : unknown error 메시지
  헤더 파일
    <string.h>

오류 번호로 오류 원인 얻기

19-9

```
#include <stdio.h>
#include <string.h>   /* strerror 함수를 정의하는 헤
      더 파일 */

main()
{
   FILE *fp;

   /* fopen 함수 호출에 실패하면 NULL이 반환 */
   if ((fp=fopen("nodata", "r")) == NULL) {
      /* errno를 설명하는 문자열을 표준 오류인 모니
       터로 출력 */
      fprintf(stderr, "ERROR: %s₩n", strerror(errno));
      exit(1);
   }
   exit(0);
}
```

19-10

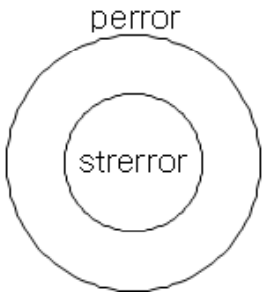
```
#include <stdio.h>
#include <string.h>   /* strerror 함수를 정의하는 헤
      더 파일 */

main()
{
   int i;

   /* 오류 번호 1-124에 대한 설명 출력 */
   for (i=0; i<125; i++)
      printf("[%3d] %s₩n", i, strerror(i));
   exit(0);
}
```

오류 메시지 출력하기

- perror 함수
  기능
    오류 메시지를 출력한다.
  기본형
    void perror(const char *s);
    s : 오류 발생시 오류 메시지 앞부분에 출력할 문자열
  반환값
    없음
  헤더 파일
    <stdio.h>

오류 메시지 출력하기

- perror 함수에서 strerror 함수 이용




```
perror("ERROR");

fprintf(stderr, "ERROR: %s₩n", strerror(errno
    ));

my_perror(char *str)
{
fprintf(stderr, "%s: %s₩n", str, strerror(errno
    ));
}
```

- 19-11

```
#include <stdio.h>

main()
{
  FILE *fp;

  if ((fp=fopen("nodata", "r")) == NULL) { /* fopen
     함수 호출 오류 발생하면 */
    perror("ERROR");
    exit(1);
  }
  exit(0);
}
```