Chapter 10: Realtime extensions (2/3)

김선영
sunyzero@gmail(dot)com
버 전: 2017-05-27

가메 출판사 http://www.kame.co.kr
저자홈페이지 http://sunyzero.tistory.com

AIO

- Asynchronous vs Non-blocking
  - Asynchronous : non-sequential & background
  - non-blocking : sequential & foreground
  - Async와 nonblocking은 의외로 많이 헷갈린다.
    - async는 순서가 기준이고, nonblocking은 함수 호출 후 즉각 리턴인지 아닌지의 여부이다. 둘은 전혀 다른 비교할 수 있는 개념이 아니다.
- For Disk I/O (not for socket) - Linux specific

AIO API list

입출력	**aio_read(3)**	비동기화된 입력을 요청합니다.
	aio_write(3)	비동기화된 출력을 요청합니다.
	lio_listio(3)	비동기 **I/O** 리스트를 처리합니다.
확인	**aio_return(3)**	비동기 입출력 요청의 결과(리턴상태)를 보고합니다.
	aio_error(3)	비동기 입출력중에 에러발생을 확인합니다.
집행	**aio_fsync(3)**	동기화를 위하여 비동기적 버퍼를 비웁니다.
	aio_suspend(3)	비동기화된 입력이 처리되기를 기다립니다. (타임아웃지정가능)
	aio_cancel(3)	아직 미해결된 비동기 입출력 요청을 취소합니다.

aiocb

- Async. I/O Control block

```
struct aiocb {
    int             aio_fildes;     /* 파일 기술자 */
    off_t           aio_offset;     /* I/O 를 처리할 파일 오프셋 */
    volatile void  *aio_buf;        /* 버퍼 위치 */
    size_t          aio_nbytes;     /* 버퍼 길이 */
    int             aio_reqprio;    /* 요청된 우선순위 */
    struct sigevent aio_sigevent;   /* 발생시킬 시그널 구조체 */
    int             aio_lio_opcode;  /* listio 작동 코드 */
    int             aio_flags;       /* flags */
};
```

AIO API prototype

- return value: aio_error (EINPROGRESS ... 0)

```
int aio_read(struct aiocb *aiocbp);
int aio_write(struct aiocb *aiocbp);
ssize_t aio_return(struct aiocb *aiocbp);
int aio_error(const struct aiocb *aiocbp);
int aio_fsync(int op, struct aiocb *aiocbp);
int aio_suspend(const struct aiocb * const cblist[], int n,
          const struct timespec *timeout);
int aio_cancel(int fd, struct aiocb *aiocbp);
```

EINPROGRESS	**Inprogress**
ECANCELED	**Canceled**
EINVAL	**Invalid argument**
0	**I/O operation completed**

aio_basic.c (1/3)

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>     /* EXIT_SUCCESS, EXIT_FAILURE */
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <aio.h>
#include <errno.h>

#define FILENAME    "fd_test.log"
#define TEST_MSG    "[Test message:1234567890]\n"
```

aio_basic.c (2/3)

```
nt main() {
    int     fd, ret;
    struct aiocb aio_wb;
    if ((fd = open(FILENAME, O_CREAT|O_RDWR, 0644)) == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }
    write(fd, TEST_MSG, sizeof(TEST_MSG));  /* synchronous I/O API */

    memset(&aio_wb, 0, sizeof(struct aiocb));
    aio_wb.aio_fildes = fd;             /* operated file descript   */
    aio_wb.aio_buf = TEST_MSG;          /* buffer to write */
    aio_wb.aio_nbytes = sizeof(TEST_MSG);   /* number of bytes */
    aio_wb.aio_offset = 5;          /* start offset to write */
    aio_write(&aio_wb);             /* asynchronous I/O API */
    while ((ret = aio_error(&aio_wb)) != 0) {
        if (ret == EINPROGRESS) {
            printf("aio_write has not been completed. sleep(1)\n");
            sleep(1);
        } else {
            printf("Error: aio_error = %d\n", ret);
            break;
        }
    } /* end : while */
```

aio_basic.c (3/3)

```
 if ((ret = aio_return(&aio_wb)) == -1) {
        fprintf(stderr, "Err(aio_write) : %s\n", strerror(errno));
    }
    printf("> write complete: %d B\n", ret);

    fsync(fd);
    close(fd);

    return EXIT_SUCCESS;
} /* end : main */
```

list AIO

- notification method : sigevent

```
int lio_listio(int mode,
               struct aiocb *restrict const list[restrict],
               int nent, struct sigevent *restrict sig);
```

- mode

LIO_WAIT	**Blocking**
LIO_NOWAIT	**Non-blocking (notification: sigevent)**

aio_listio.c (1/5)

```
#define NUM_AIOCB_ARRAY     5

#ifdef ASYNCHRONIZED_LIO
#include <pthread.h>
void start_sigev_aio(sigval_t   arg);

pthread_attr_t  pt_attr;
pthread_barrier_t   pt_barrier; /* 2014-08-16 */
#endif

int main(int argc, char *argv[]) {
    int     fd_rd, fd_wr;
    int     i =0, tot_len = 0, ret;
    char    ofname[0xff], rbuf[NUM_AIOCB_ARRAY][200000];
    struct aiocb  aio_blk[NUM_AIOCB_ARRAY];
    struct aiocb  *aiolist[] = {
        &aio_blk[0], &aio_blk[1],
        &aio_blk[2], &aio_blk[3], &aio_blk[4] };

#ifdef ASYNCHRONIZED_LIO
    struct sigevent sigev = { .sigev_notify = SIGEV_THREAD };
#endif
```

aio_listio.c (2/5)

```
if (argc != 2) {
    printf("Usage : %s <source file>\n", argv[0]);
    return EXIT_SUCCESS;
}
sprintf(ofname, "%s.copy", argv[1]);
printf("file copy : %s =>>> %s\n", argv[1], ofname);
if ((fd_rd = open(argv[1], O_RDONLY, 0)) == -1) {
    perror("Fail: open()");
    exit(EXIT_FAILURE);
}
if ((fd_wr = open(ofname, O_CREAT|O_WRONLY|O_TRUNC, 0644)) == -1){
    perror("Fail: open()");
    exit(EXIT_FAILURE);
}

memset(aio_blk, 0, sizeof(struct aiocb) * NUM_AIOCB_ARRAY);
```

aio_listio.c (3/5)

```
while(1) {
        ret = read(fd_rd, rbuf[i], sizeof(rbuf[i]));
        if (ret == -1) { /* error */
            exit(1);
        }

        aio_blk[i].aio_fildes = fd_wr;  /* file descriptor */
        aio_blk[i].aio_buf = rbuf[i];   /* buffer to write */
        aio_blk[i].aio_nbytes = ret;    /* number of bytes */
        aio_blk[i].aio_offset = tot_len;/* start offset    */
        aio_blk[i].aio_lio_opcode = LIO_WRITE; /* operation mode */

        /* update conditions */
        tot_len += ret;
        i++;
```

posix_fadvise를 사용하도록 수정해보자.

= 이것은 여러분의 연습을 위해 남겨두겠다.
절대로 귀찮아서가 아니다.

aio_listio.c (4/5)

```
        if (i==NUM_AIOCB_ARRAY || ret == 0) { /* the time to write */
#ifdef ASYNCHRONIZED_LIO
            pthread_barrier_init(&pt_barrier, NULL, 2);

            sigev.sigev_notify = SIGEV_THREAD;
            sigev.sigev_value.sival_int = tot_len; /* thread argument */

            pthread_attr_init(&pt_attr); /* thread attribute */
            pthread_attr_setdetachstate(&pt_attr, PTHREAD_CREATE_DETACHED);
            sigev.sigev_notify_attributes = &pt_attr;
            sigev.sigev_notify_function = start_sigev_aio;
            lio_listio(LIO_NOWAIT, aiolist, ret == 0 ? i-1:i, &sigev);
            pthread_barrier_wait(&pt_barrier);   /* barrier (2014-08-16) */
            pthread_barrier_destroy(&pt_barrier);//destory barrier before re-init
#else
            lio_listio(LIO_WAIT, aiolist, ret == 0 ? i-1:i, NULL);
#endif
            memset(aio_blk, 0, sizeof(struct aiocb) * NUM_AIOCB_ARRAY);
            i=0; /* reset index */
        } /* end : if */
```

5번(NUM_AIOCB_ARRAY)마다
스레드 배리어를 이용해서 대기하도록 한다.
이는 AIOCB가 오염되는 것을 막는다.

aio_listio.c (5/5)

```
        if (ret == 0) break; /* EOF then break */
    } /* loop: while */

    printf("> write complete\n");
    close(fd_rd);

    fsync(fd_wr);
    close(fd_wr);
    return (EXIT_SUCCESS);
}

#ifdef ASYNCHRONIZED_LIO
void start_sigev_aio(sigval_t arg)
{
    printf("[SIGEV] thread(%ld) by lio_listio (len:%d).\n",
        (long)pthread_self(), arg.sival_int);
    pthread_barrier_wait(&pt_barrier);    /* barrier (2014-08-16) */
}
#endif
```

AIO SIGEV로 호출될 스레드 함수이다.

Test : AIO, LIO, cp

```
$ time cp ~/debian-506-i386-DVD-1.iso .
real    1m39.689s
user    0m0.123s
sys     0m27.509s
```

```
$ time ./aio_listio ~/debian-506-i386-DVD-1.iso
file copy : /home/sunyzero/debian-506-i386-DVD-1.iso -> ..생략..copy
> write complete

real    1m22.598s
user    0m0.258s
sys     0m20.911s
```

```
$ time ./aio_listio ~/debian-506-i386-DVD-1.iso
file copy : /home/sunyzero/debian-506-i386-DVD-1.iso -> ..생략..copy
> write complete

real    1m14.488s
user    0m0.269s
sys     0m20.394s
```