# 10주차 3차시 메모리 관련 함수

## 【학습목표】

1. 데이터 교환을 설명할 수 있다.
2. 데이터 교환 함수를 사용할 수 있다.

## 학습내용1 : 메모리 매핑 활용 데이터 교환

메모리 매핑을 이용한 데이터 교환은 부모 프로세스와 자식 프로세스가 메모리 매핑을 사용하여 데이터 교환 가능

### 1. 매핑된 메모리 동기화

- 매핑된 메모리의 내용과 백업 내용이 일치하도록 동기화 필요

* 매핑된 메모리 동기화: msync(3)
addr로 시작하는 메모리 영역에서 len 길이만큼의 내용을 백업저장장치에 기록

```
#include <sys/mman.h>

int msync(void *addr, size_t len, int flags);
```

addr : 매핑된 메모리의 시작 주소
len : 메모리 영역의 크기
flags : 동기화 동작

* flags : 함수의 동작 지시
MS_ASYNC : 비동기 쓰기 작업
MS_SYNC : 쓰기 작업을 완료할 때까지 msync 함수는 리턴 안함
MS_INVALIDATE : 메모리에 복사되어 있는 내용을 무효화

* msync 함수 사용하기(1)

```
...
08  int main(int argc, char *argv[]) {
09      int fd;
10      caddr_t addr;
11      struct stat statbuf;
12
13      if (argc != 2) {
14          fprintf(stderr, "Usage : %s filename\n", argv[0]);
15          exit(1);
16      }
17
18      if (stat(argv[1], &statbuf) == -1) {
19          perror("stat");
20          exit(1);
21      }
22
23      if ((fd = open(argv[1], O_RDWR)) == -1) {
24          perror("open");
25          exit(1);
26      }
28      addr = mmap(NULL, statbuf.st_size, PROT_READ|PROT_WRITE,
29                  MAP_SHARED, fd, (off_t)0);
30      if (addr == MAP_FAILED) {
31          perror("mmap");
32          exit(1);
33      }
34      close(fd);
35
36      printf("%s", addr);
37
38      printf("---------\n");
39      addr[0] = 'D';
40      printf("%s", addr);
41
42      msync(addr, statbuf.st_size, MS_SYNC);
43
44      return 0;
45  }
```

파일의 상세 정보 검색

메모리 매핑

매핑된 내용 출력

매핑된 내용 수정

수정된 내용 동기화

```
# cat mmap.dat
HANBIT
BOOK
# ex8_4.out mmap.dat
HANBIT
BOOK
---------
DANBIT
BOOK
# cat mmap.dat
DANBIT
BOOK
```

## 2. 데이터 교환하기

- 부모 프로세스나 자식 프로세스가 매핑된 메모리의 내용을 변경하면 다른 프로세스도 변경 내용을 알 수 있다.

```
...
09  int main(int argc, char *argv[]) {
10      int fd;
11      pid_t pid;
12      caddr_t addr;
13      struct stat statbuf;
14
15      if (argc != 2) {
16          fprintf(stderr, "Usage : %s filename\n", argv[0]);
17          exit(1);
18      }
19
20      if (stat(argv[1], &statbuf) == -1) {
21          perror("stat");
22          exit(1);
23      }
24
25      if ((fd = open(argv[1], O_RDWR)) == -1) {
26          perror("open");
27          exit(1);
28      }
29
30      addr = mmap(NULL, statbuf.st_size, PROT_READ|PROT_WRITE,
31                  MAP_SHARED, fd, (off_t)0);
32      if (addr == MAP_FAILED) {
33          perror("mmap");
34          exit(1);
35      }
36      close(fd);
37
38      switch (pid = fork()) {
39          case -1 :  /* fork failed */
40              perror("fork");
41              exit(1);
42              break;
43          case 0 :   /* child process */
44              printf("1. Child Process : addr=%s", addr);
45              sleep(1);
46              addr[0] = 'x';
47              printf("2. Child Process : addr=%s", addr);
48              sleep(2);
49              printf("3. Child Process : addr=%s", addr);
50              break;
51          default :  /* parent process */
52              printf("1. Parent process : addr=%s", addr);
53              sleep(2);
54              printf("2. Parent process : addr=%s", addr);
55              addr[1] = 'y';
56              printf("3. Parent process : addr=%s", addr);
57              break;
58      }
59
60      return 0;
61  }
```

메모리 매핑

fork 함수로 자식 프로세스 생성

자식 프로세스가 매핑된 내용 수정

부모 프로세스가 매핑된 내용 수정

```
# cat mmap.dat
HANBIT BOOK
# ex8_5.out mmap.dat
1. Child Process : addr=HANBIT BOOK
1. Parent process : addr=HANBIT BOOK
2. Child Process : addr=xANBIT BOOK
2. Parent process : addr=xANBIT BOOK
3. Parent process : addr=xyNBIT BOOK
3. Child Process : addr=xyNBIT BOOK
# cat mmap.dat
xyNBIT BOOK
#
```

| 시간 | Child process | Parent process |
|---|---|---|
| 0 | printf("1. ") | printf("1. ") |
| 1 | addr[0]='x'<br>printf("2. ") | |
| 2 | | printf("2. ")<br>Addr[1]='y' |
| 3 | printf("3. ") | printf("3. ") |

## 학습내용2 : 데이터 교환 함수

### 1. 메모리 매핑 함수

| 기능 | 함수원형 |
|---|---|
| 메모리 매핑 | void *mmap(void *addr, size_t len, int prot, int flags, int fildes, off_t off); |
| 메모리 매핑 해제 | int munmap(void *addr, size_t len); |
| 파일 크기 조정 | int truncate(const char *path, off_t length);<br>int ftruncate(int fildes, off_t length); |
| 매핑된 메모리 동기화 | int msync(void *addr, size_t len, int flages); |

**【학습정리】**

1. 메모리 매핑 활용 데이터 교환
메모리 매핑을 이용한 데이터 교환은 부모 프로세스와 자식 프로세스가 메모리 매핑을 사용하여 데이터 교환 가능

2. 매핑된 메모리 동기화
- 매핑된 메모리의 내용과 백업 내용이 일치하도록 동기화 필요
- 매핑된 메모리 동기화: msync(3)

3. 데이터 교환 함수

| 기능 | 함수원형 |
|---|---|
| 메모리 매핑 | void *mmap(void *addr, size_t len, int prot, int flags, int fildes, off_t off); |
| 메모리 매핑 해제 | int munmap(void *addr, size_t len); |
| 파일 크기 조정 | int truncate(const char *path, off_t length);<br>int ftruncate(int fildes, off_t length); |
| 매핑된 메모리 동기화 | int msync(void *addr, size_t len, int flages); |