

UNIX 시스템 프로그래밍

» 10장. 시스템 V의 프로세스간 통신

shared memory

- ▶ 둘 이상의 프로세스가 물리적 메모리의 일부를 공유
- ▶ 가장 효율적인 IPC 기법

shmget 시스템 호출

▶ 사용법 :

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

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

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

```
int shmget(key_t key, size_t size, int permflag);
```

- key : 공유 메모리 영역의 identifier
- size : 공유 메모리 영역의 최소 크기
- permflag : access permission|IPC_CREAT|IPC_EXCL
- return 값 : 공유 메모리 영역의 identifier

공유 메모리 생성 예

- ▶ 512byte의 문자를 저장 할 공유 메모리 생성
 - `shmid1=shmget(0111, 512, 0600|IPC_CREAT);`
- ▶ 10개의 정수를 저장 할 공유 메모리 생성
 - `shmid2=shmget(0112, 10*sizeof(int), 0600|IPC_CREAT);`
- ▶ struct databuf의 데이터 5개를 저장 할 공유 메모리 생성
 - `shmid3=shmget(0113, 5*sizeof(struct databuf), 0600|IPC_CREAT);`

shmat 시스템 호출

- ▶ shmget 호출에 의해 할당된 메모리 영역을 자신의 논리적 자료 공간에 부착
- ▶ 사용법 :

```
#include <sys/shm.h>
int *shmat (int shmid, const void *daddr, int shmflag);
```

 - shmid : 공유 메모리 identifier

shmat 시스템 호출 (2)

- daddr :
 - process address space 내의 부착 위치
 - NULL 인 경우 시스템이 위치 결정
- shmflag :
 - SHM_RDONLY : 공유 메모리에 대해 읽기만 가능
 - 0 : 공유 메모리에 대해 읽기/쓰기 가능
- return 값 : process 내의 유효주소
 - 실패 시 : (void *) -1

shmdt 시스템 호출

- ▶ 공유메모리 영역을 프로세스의 논리적 주소 공간으로부터 떼어낸다.
- ▶ 사용법 :
 int shmdt (memptr);
 - memptr : 공유 메모리 영역에 대한 유효주소
 - return 값 : 0 or -1

shmat를 이용한 공유 메모리 부착 예

- ▶ 512byte의 문자를 저장 할 공유 메모리 생성 후 부착
 - `buf1=(char *)shmat(shmid1,0,0);`
- ▶ 10개의 정수를 저장 할 공유 메모리 생성 후 부착
 - `buf2=(int *)shmat(shmid2,0,0);`
- ▶ struct databuf의 데이터 5개를 저장 할 공유 메모리 생성 후 부착
 - `buf3=(struct databuf *)shmat(shmid3,0,0);`

공유 메모리 사용 예

- ▶ 표준 입력으로 읽은 문자열을 공유 메모리 공간에 저장 후 출력

```
n=read(0, buf1, 512);  
write(1, buf1, n);
```

- ▶ 표준 입력으로 읽은 10개의 정수를 공유 메모리 공간에 저장 후 출력

```
for (i=0; i<10; i++)  
    scanf("%d", buf2+i);
```

```
for (i=0; i<10; i++)  
    printf("%d\n", *(buf2+i));
```

공유 메모리 사용 예 (2)

- ▶ struct databuf의 데이터 중 d_nread에 10씩 더하기

```
for (i=0; i<5; i++)  
    (buf3+i)->d_nread+=10;
```

- ▶ struct databuf의 데이터 중 d_nread와 d_buf 출력하기

```
for (i=0; i<5; i++)  
    printf("%d ... %s", (buf3+i)->d_nread, (buf3+i)->d_buf);
```

shmctl 시스템 호출

- ▶ 사용법 :

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

```
int shmctl (int shmid, int command, struct  
shm_id_ds *shm_stat);
```

- ▶ command

- IPC_STAT
- IPC_RMID

record locking

▶ 필요성 :

현재 : $X=100$

P1
Read X
 $X=X+100$
Write X

P2
Read X
 $X=X+200$
Write X

P1과 P2의 실행 후 X의 값은 ?

record locking (2)

- ▶ locking : 특정 record에 대한 다른 프로세스의 읽기/쓰기 제한
 - read lock : 읽기는 허용, 쓰기는 제한
 - write lock : 읽기, 쓰기 모두 제한
- ▶ unlocking : 제한 해제

record locking (3)

▶ 사용법 :

```
#include <fcntl.h>
```

```
int fcntl(int filedes, int cmd, struct flock *ldata);
```

- filedes : lock을 설정 하려는 file의 descriptor
 - read-lock은 O_RDONLY/O_RDWR로 open된 file에 한해서 적용 가능
 - write-lock은 O_WRONLY/O_RDWR로 open된 file에 한해서 적용 가능

record locking (4)

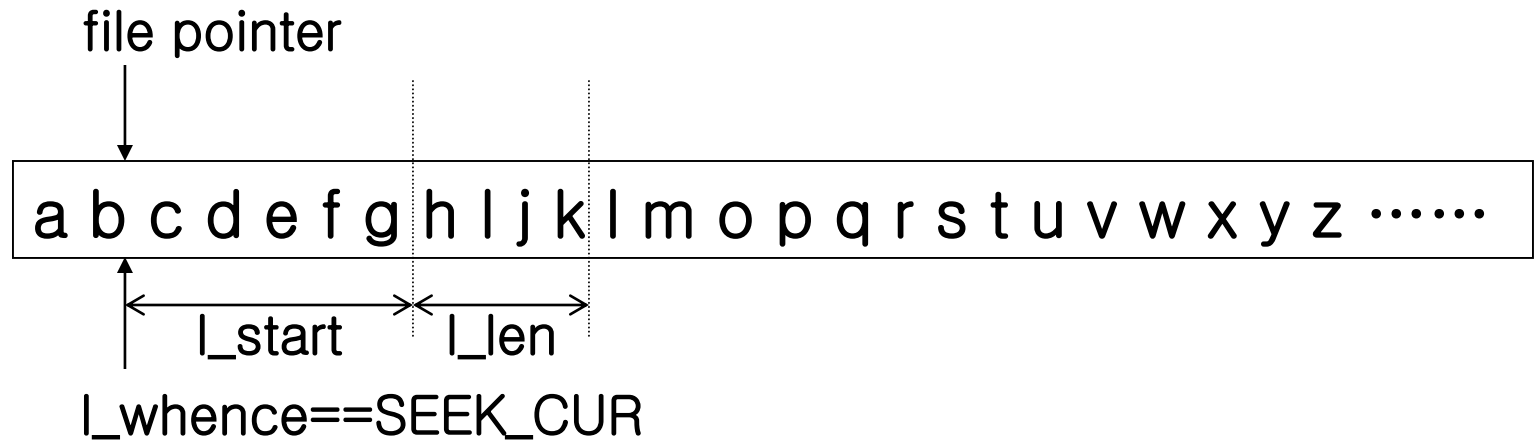
- cmd=

- F_GETLK : lock 정보 얻기
 - 해당 정보는 세 번째 인수에 저장
- F_SETLK : non-blocking locking or unlocking
 - lock 설정에 관한 자세한 정보는 세 번째 인수에 지정
- F_SETLKW : blocking locking
 - lock 설정에 관한 자세한 정보는 세 번째 인수에 지정

record locking (5)

- struct flock *ldata
 - short l_type : lock의 type
 - F_RDLCK, F_WRLCK, F_UNLCK
 - short l_whence :
 - SEEK_SET, SEEK_CUR, SEEK_END
 - off_t l_start : l_whence로 부터의 변위로 표현된 locked record의 시작 위치
 - off_t l_len : locked record의 길이
 - pid_t l_pid : F_GETLK의 경우만 유효
 - 누가 해당 file에 lock을 걸었나?

record locking (6)



- ▶ lock 정보는 `fork()`에 의해 계승되지 않는다.
- ▶ 모든 lock은 프로세스 종료 시 자동으로 unlock 된다.

교착상태 (Deadlock)

- ▶ 교착 상태란 ?
- ▶ 교착상태 검색 가능 ?
 - F_SETLKW 명령에 대해 -1 return
 - errno는 EDEADLK

locking의 사용 예

```
int main(void){
    int fd, i, num;
    struct flock lock;

    fd=open("data1", O_RDWR|O_CREAT, 0600);

    lock.l_whence=SEEK_CUR;
    lock.l_len=sizeof(int);

    for (i=0; i<10; i++){
        lock.l_type=F_WRLCK;
        lock.l_start=0;
        fcntl(fd, F_SETLKW, &lock);

        read(fd, &num, sizeof(int));
        num=num+10;
        sleep(1);
    }
}
```

locking의 사용 예 (2)

```
lseek(fd, -sizeof(int), SEEK_CUR);  
write(fd, &num, sizeof(int));
```

```
lock.l_type=F_UNLCK;  
lock.l_start=-sizeof(int);  
fcntl(fd, F_SETLK, &lock);
```

```
}
```

```
lseek(fd, 0, SEEK_SET);  
for (i=0; i<10; i++){  
    read(fd, &num, sizeof(int));  
    printf("%d\n", num);
```

```
}
```

```
return 0;
```

```
}
```

