강원혁신플랫폼

리눅스프로그래밍

공유 메모리

Linux 시스템에서 여러 프로세스가 공통 메모리 세그먼트에 액세스하고 공유할 수 있도록 하는 메커니즘을 무엇이라고 하나요

공유 메모리

학습목표

학습 내용

1 공유 메모리의 주요 함수

학습 목표

공유 메모리의 주요 함수를 파악할 수 있다.

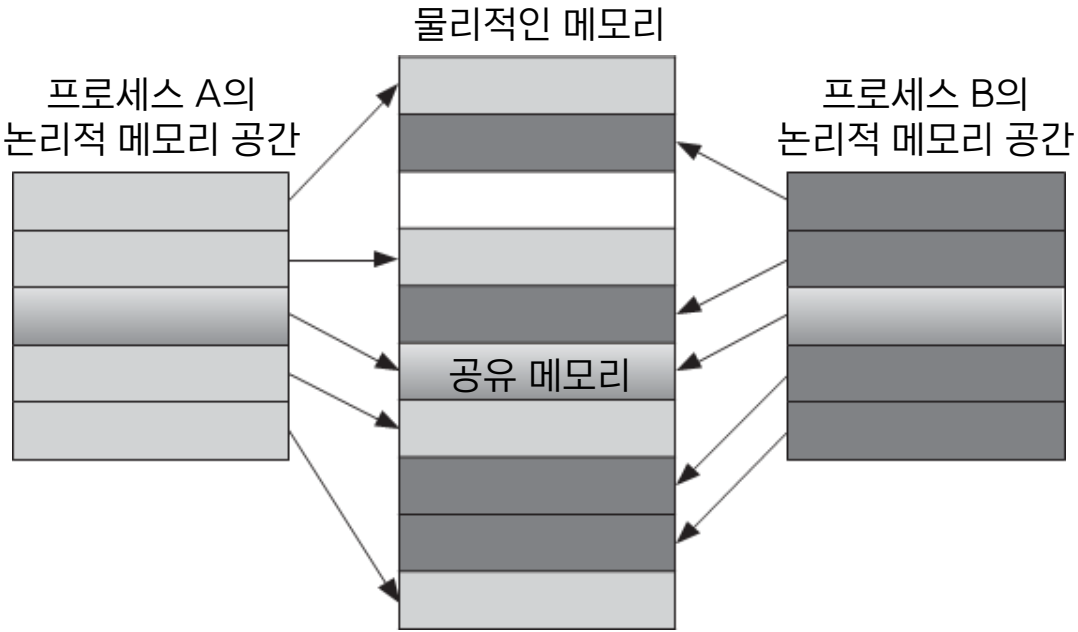
공유 메모리의 주요 함수

공유 메모리

특징

- 여러 프로세스가 공통 메모리 세그먼트에 액세스하고 공유할 수 있도록 하는 메커니즘
- 프로세스가 데이터를 복사할 필요 없이 공유 메모리에서 직접 읽고 쓸 수 있으므로 프로세스 간 통신(IPC)의 빠르고 효율적인 수단 제공

공유 메모리



출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

공유 메모리 semid_ds

공유 메모리 semid_ds 구조체를 사용

```
struct shmid_ds {
    struct ipc_perm shm_perm;    /* Ownership and permissions */
    size_t          shm_segsz;   /* Size of segment (bytes) */
    time_t          shm_atime;   /* Last attach time */
    time_t          shm_dtime;   /* Last detach time */
    time_t          shm_ctime;   /* Last change time */
    pid_t           shm_cpid;    /* PID of creator */
    pid_t           shm_lpid;    /* PID of last shmat(2)/shmdt(2) */
    shmatt_t        shm_nattch;  /* No. of current attaches */
    ...
};
```

공유 메모리의 주요 함수

shmget(): 새로운 공유 메모리 세그먼트를 생성하거나
기존 공유 메모리 세그먼트의 식별자를 얻는 데 사용

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

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

공유 메모리의 주요 함수

shmat(): 공유 메모리 세그먼트를 프로세스의 주소 공간에 연결하여
프로세스가 공유 메모리에 액세스하고 조작할 수 있도록 하는 데 사용

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

void *shmat(int shmid, const void *shmaddr, int shmflg);
```

공유 메모리의 주요 함수

shmdt(): 프로세스의 주소 공간에서 공유 메모리 세그먼트를 분리하여
프로세스가 더 이상 공유 메모리에 액세스할 필요가 없음을 나타냄

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

int shmdt(const void *shmaddr);
```

공유 메모리의 주요 함수

shmctl(): 공유 메모리 세그먼트를 제어하고 관리하는 데 사용

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

int shmctl(int shmid, int cmd, struct shmid_ds *buf);
```

따라하기 shm.c


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

#define SHM_KEY 0x12345 /* 공유 메모리 키 */

int main(int argc, char **argv) {
    int i, pid, shmid;
    int *cVal;
    void *shmmem = (void *)0;
```

따라하기 shm.c


```
if((pid = fork()) == 0) {/* 자식 프로세스 */
        /* 공유 메모리 공간을 가져온다. */
        shmid = shmget((key_t)SHM_KEY, sizeof(int), 0);
        if(shmid == -1) {
            perror("shmget()");
            return -1;
        }

/* 공유 메모리를 사용하기 위해 프로세스의 메모리에 붙인다. */
        shmmem = shmat(shmid, (void *)0, 0666 | IPC_CREAT);
        if(shmmem == (void *)-1) {
            perror("shmat()");
```

```
return -1;
        }
        cVal = (int *)shmmem;
        *cVal = 1;
        for(i = 0; i < 3; i++) {
            *cVal += 1;
            printf("Child(%d) : %d\n", i, *cVal);
            sleep(1);
        }
    }
```

따라하기 shm.c


```
} else if(pid > 0) { /* 부모 프로세스, 공유 메모리 내용 표시 */
        /* 공유 메모리 공간을 만든다. */
        shmid = shmget((key_t)SHM_KEY, sizeof(int), 0666 | IPC_CREAT);
        if(shmid == -1) {
            perror("shmget()");
            return -1;
        }

/* 공유 메모리를 사용하기 위해 프로세스의 메모리에 붙인다. */
        shmmem = shmat(shmid, (void *)0, 0);
        if(shmmem == (void *)-1) {
            perror("shmat()");
```

```
return -1;
        }

        cVal = (int *)shmmem;
        for(i = 0; i < 3; i++) {
            sleep(1);
            printf("Parent(%d) : %d\n", i, *cVal);
        }
    }
    shmctl(shmid, IPC_RMID, 0);
    return 0;
}
```

따라하기 shm.c

- 실행결과
  - 자식 ps에서는 변수값 증가, 화면에 출력, 1초간 슬립
  - 부모 ps에서는 1초 슬립후 변수값을 읽어 출력함

```
$ ./shm
Child(0) : 2
Parent(0) : 2
Child(1) : 3
Parent(1) : 3
Child(2) : 4
Parent(2) : 4
$
```

01 • 공유 메모리의 주요 함수

<sys/shm.h>
shmget( )
shmctl( )
shmat( ), shmdt( )