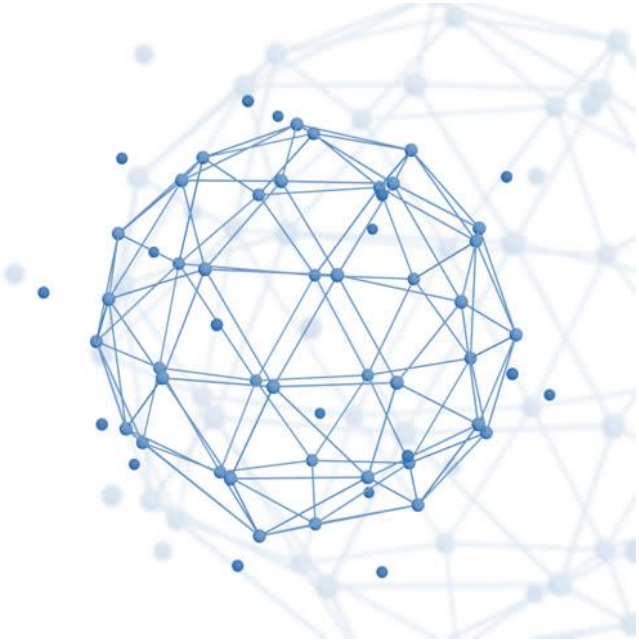
HALLYM UNIVERSITY

Shared memory & Socket

2021.06.02

Sang-woo Lee

glutton.leesw@gmail.com



CIC Convergence Information & Communications

Contents

- Shared memory
- Socket

Shared memory

- Shared memory
  - 일반적으로 메모리는 하나의 프로세서만 사용함.
  - 여러 프로세서에서 메모리를 공유하여 사용하는 것이 Shared memory.
- Shared memory 사용을 위한 함수

int shmget(key_t key, size_t size, int shmflg)
<설명> key의 값으로 공유메모리를 얻고, 공유메모리 조각(shared memory segment)의 id를 반환함.
<인자 값> 1. key : 공유메모리를 할당할 때 사용하는 고유 key 값. 2. 메모리의 최소 size. 만약 이미 존재하는 메모리라면 0으로 표기함. 3. 옵션flag: **IPC_CREAT** → 새로운 메모리 세그먼트를 생성 (덮어쓰기) **IPC_EXCL** → 만약 기존 공유 메모리 세그먼트가 존재하면 shmget은 실패 (-1 반환)

void *shmat(int shmid, const void *shmaddr, int shmflg);
<설명> 공유 메모리 id를 사용해서 현재 프로세스와 공유메모리를 연결함 (attach)
<인자 값> 1. shmid: 공유 메모리의 id. 2. shmaddr: NULL(0) → 공유메모리의 주소를 반환. *주의) 공유메모리의 주소를 가리키는 void 포인터를 반환함.

Shared memory

- Shared memory 사용을 위한 함수

int shmdt(const void *shmaddr);
<설명> 현재 프로세스와 공유메모리의 연결을 해제함 (detach)
<인자 값> 1. shmid: **shmat**에서 전달받은 그 포인터

int shmctl(int shmid, int cmd, struct shmid_ds *buf);
<설명> 공유메모리를 제어함 (삭제, 정보 조회 등)
<인자 값> 1. shmid: **shmat**에서 전달받은 그 포인터 2. cmd : 명령어 → 정수형을 갖으며 여러 command를 통해 다양한 작업 수행 (삭제, 정보 조회 등) 3. Buf : shmid_ds라는 구조체로 정의되어 있음. 공유메모리의 정보를 담는 구조체

Shared memory

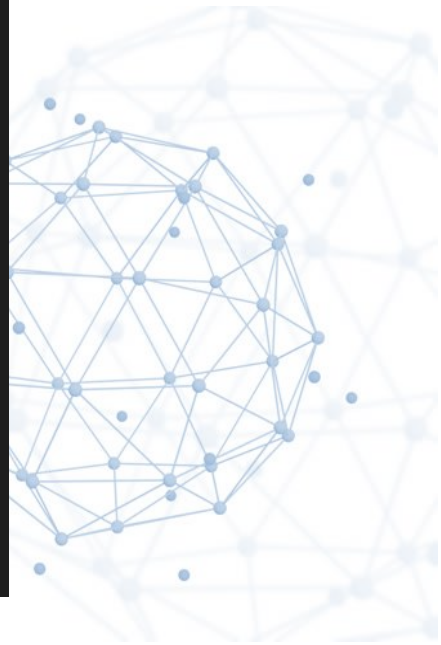
- shared_memory1.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/shm.h>
#include <sys/ipc.h>
#include <signal.h>

#define SIZE 1024

void signalHandler(int signo);
int shmid;

int main (int argc, char *argv[]){
    void *shmaddr; // void형 포인터
    shmid = shmget( (key_t)1234, SIZE, IPC_CREAT|0666 );
    // shmget -> 키값에 해당되는 shared memory를 얻고, shared memory segment의 id를 반환.
    if(shmid==-1){
        printf("shmget error\n");
        exit(1);
    }

    shmaddr=shmat(shmid, (void *)0, 0);
    // shmat -> 이 프로세스를 shared memory segment와 묶어줌(attach)
    // 성공하면 공유메모리에 접근할 수 있는 포인터 값을 반환함.
    if(shmaddr==(void *)-1){
        printf("shmaddr error\n");
        exit(1);
    }

```

Shared memory

- shared_memory1.c

```
32      strcpy( (char *)shmaddr, "Linux Programming");
33      // 포인터를 사용하여, 공유메모리에 데이터 저장.
34
35      if (shmdt(shmaddr) == -1){
36          // shmdt -> 이 프로세스를 shared memory segment와 분리함(dettach)
37          printf("shmdt error\n");
38          exit(1);
39      }
40
41      // 시그널 대기
42      signal(SIGINT, signalHandler);
43      pause();
44      return 0;
45  }
46
47  void signalHandler(int signo){
48      if(shmctl(shmid, IPC_RMID, 0) == -1){
49          // shmctl는 공유메모리를 제어함, IPC_RMID는 공유메모리 삭제 command
50          printf("shmctl error\n");
51          exit(1);
52      }
53      exit(0);
54  }
```



Shared memory

- shared_memory2.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/shm.h>
#include <sys/ipc.h>
#include <signal.h>

#define SIZE 1024

int main(){
    int shmid;
    void *shmaddr;
    struct shmid_ds shm_stat; // sys/shm.h에 저장된 구조체, shared memory의 정보를 담고 있음.

    shmid = shmget( (key_t)1234, SIZE, IPC_CREAT|0666 );
    // shmget -> 키값에 해당되는 shared memory를 얻고, shared memory segment의 id를 반환.
    if(shmid==-1){
        printf("shmget error\n");
        exit(1);
    }

    shmaddr=shmat(shmid, (void *)0, 0);
    // shmat -> 이 프로세스를 shared memory segment와 묶어줌(attach)
    // 성공하면 공유메모리에 접근할 수 있는 포인터 값을 반환함.
    if(shmaddr==(void *)-1){
        printf("shmaddr error\n");
        exit(1);
    }
    printf("data read from shared memory : %s\n", (char *)shmaddr);

```

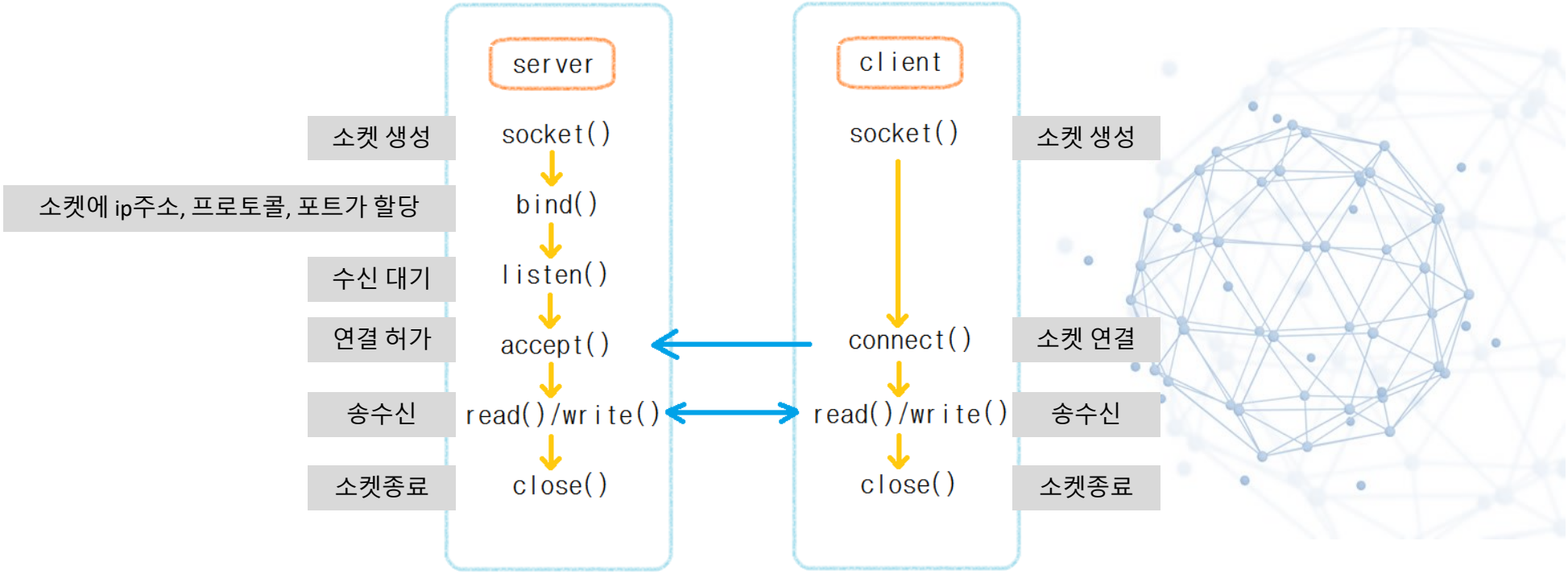
Shared memory

- shared_memory2.c

```
32
33      if(shmctl(shmid, IPC_STAT, &shm_stat) == -1){
34          // shmctl는 공유메모리를 제어함, IPC_STAT는 공유메모리 정보를 세 번째 인자값에 저장
35          printf("shmctl error\n");
36          exit(1);
37      }
38
39      if (shmdt(shmaddr) == -1){
40          // 프로세스를 공유메모리와 분리
41          printf("shmdt error\n");
42          exit(1);
43      }
44      kill(shm_stat.shm_cpid, SIGINT);
45  }
```

Socket

- Socket
  - 원래 뜻은 특정 파이프의 끝.
  - 네트워크 혹은 인터넷 통신의 종착점 (end-point)를 말함.
  - 네트워크 망을 구성하는 다양한 통신 프로토콜에 대한 이해 없이, end-point로서 연결-데이터 송수을 가능하게 함.
  - Socket을 사용한 프로세스간 통신 가능.



Socket

- socket_server.c

```c
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>

void child_proc(int conn);

int main(int argc, char const *argv[]){
  int listen_fd, new_socket ;
  struct sockaddr_in address;
  int opt = 1;
  int addrlen = sizeof(address);

  char buffer[1024] = {0};

  listen_fd = socket(AF_INET /*프로토콜 패밀리 선택*/, SOCK_STREAM /*소켓 타입*/, 0 /*IP*/) ; // 소켓 생성
  if (listen_fd == -1){
    perror("socket failed : ");
    exit(EXIT_FAILURE);
  }
  memset(&address, '0', sizeof(address)); // 소켓이 사용할 메모리의 크기 설정
  address.sin_family = AF_INET; // 프로토콜 패밀리
  address.sin_addr.s_addr = INADDR_ANY; // 서버 아이피 = 127.0.0.1 = local host
  address.sin_port = htons(8080); // 서버의 포트 == 8080
  if (bind(listen_fd, (struct sockaddr *)&address, sizeof(address)) ==-1) {
    // 첫번째 인자로 넘겨지는 소켓(listen_fd)과 이 프로세스와 묶는다
    perror("bind failed : ");
    exit(EXIT_FAILURE);
  }
```

Socket

- socket_server.c

```
32
33    while (1) {
34      if (listen(listen_fd, 16) == -1) {
35        //listen: 소캣을 통해 수신한다. 두번째 인자값 만큼 연결 요청을 큐에 넣는다.
36        perror("listen failed : ");
37        exit(EXIT_FAILURE);
38      }
39      new_socket = accept(listen_fd, (struct sockaddr *) &address, (socklen_t*)&addrlen) ;
40      // 새로운 연결요청이 있으면, 연결을 허락한다.
41      if (new_socket < 0) {
42        perror("accept");
43        exit(EXIT_FAILURE);
44      }
45      if (fork() > 0) {
46        //자식 프로세스면, new_socket을 통해 데이터 수신.
47        child_proc(new_socket) ;
48      }
49      else {
50        //부모 프로세스면, new_socket를 사용하지 않음..
51        close(new_socket) ;
52      }
53    }
54  }
```

Socket

- socket_server.c

```
56  void child_proc(int conn){
57    char buf[1024] ;
58    char * data = 0x0 ;
59    char * tmp ;
60    int len = 0 ;
61    int s ;
62
63    while ( (s = recv(conn, buf, 1023, 0)) > 0 ) {
64      // 데이터 수신
65      buf[s] = 0x0 ;
66      if (data == 0x0) {
67        data = strdup(buf) ;
68        len = s ;
69      }
70      else {
71        data = realloc(data, len + s + 1) ;
72        strncpy(data + len, buf, s) ;
73        data[len + s] = 0x0 ;
74        len += s ;
75      }
76    }
77    printf(">%s\n", data); // 수신한 데이터 출력
78
79    // 수신한 데이터 다시 재송신.
80    while (len > 0 && (s = send(conn, data, len, 0)) > 0) {
81      data += s ;
82      len -= s ;
83    }
84    shutdown(conn, SHUT_WR) ;
85    // 연결 종료
86  }
```

Socket

- socket_client.c

```c
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>

int main(int argc, char const *argv[]) {
  struct sockaddr_in serv_addr;
  int sock_fd ;
  int s, len ;
  char buffer[1024] = {0};
  char * data ;
  sock_fd = socket(AF_INET, SOCK_STREAM, 0); // 소켓 생성
  if (sock_fd <= 0) {
    perror("socket failed : ") ;
    exit(EXIT_FAILURE) ;
  }

  memset(&serv_addr, '0', sizeof(serv_addr)); // 소켓이 사용할 메모리의 크기 설정
  serv_addr.sin_family = AF_INET; // 프로토콜 패밀리 명시
  serv_addr.sin_port = htons(8080); // 포트 명시
  if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0) { // 서버의 IP 명시
    perror("inet_pton failed : ") ;
    exit(EXIT_FAILURE) ;
    }

  if (connect(sock_fd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0) {//연결 요청
    perror("connect failed : ") ;
    exit(EXIT_FAILURE) ;
  }
```

Socket

- socket_client.c

```
32
33      scanf("%s", buffer); // 터미널을 통해 데이터 일벽받음
34      data = buffer ;
35      len = strlen(buffer) ;
36      s = 0 ;
37      while (len > 0 && (s = send(sock_fd, data, len, 0)) > 0) {// 소켓을 통해 데이터 전송
38        data += s ;
39        len -= s ;
40      }
41
42      shutdown(sock_fd, SHUT_WR) ; // 송신을 위한 연결 종료 (수신 가능)
43
44      char buf[1024] ;
45      data = 0x0 ;
46      len = 0 ;
47      while ( (s = recv(sock_fd, buf, 1023, 0)) > 0 ) { // 데이터 수신
48        buf[s] = 0x0 ;
49        if (data == 0x0) {
50          data = strdup(buf) ;
51          len = s ;
52        }
53        else {
54          data = realloc(data, len + s + 1) ;
55          strncpy(data + len, buf, s) ;
56          data[len + s] = 0x0 ;
57          len += s ;
58        }
59
60      }
61      printf(">%s\n", data); // 수신받은 데이터 출력
62    }
```