Operating System (OS)

Lec10 : 프로세스 통신

충북대학교

강병호 (지능로봇공학과)

kang6283@chungbuk.ac.kr

01
IPC

IPC (Inter Process Communication)

- 프로세스 간 통신
- pipe, socket, shared memory ….

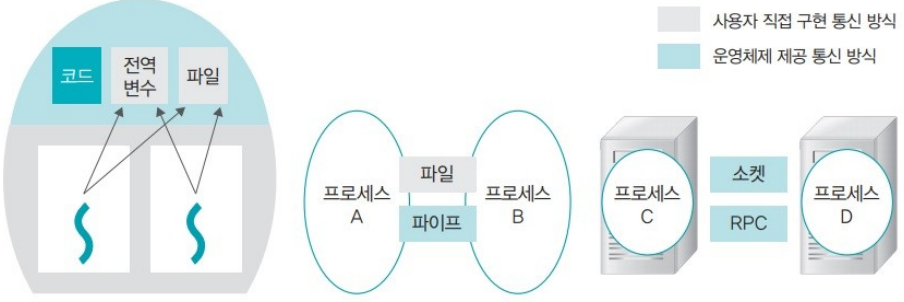
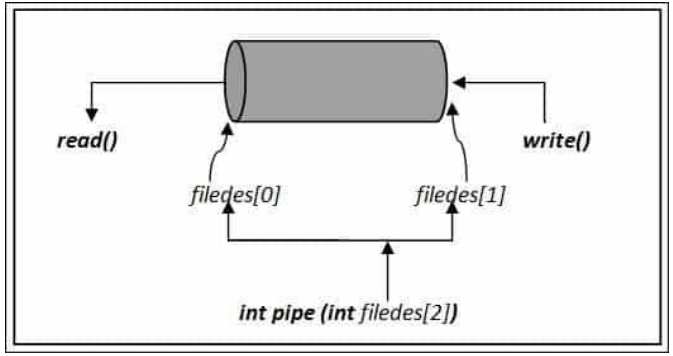


그림 5-1 **프로세스 간 통신의 종류**

02
pipe

pipe

- 두 프로세스 사이에서 한 방향으로 통신
- 파일에 쓰고 읽기와 유사한 방식
- 파일시스템이 아니라 메모리에 존재
- Anonymous pipe : 부모 – 자식 프로세스 간 통신
- Named pipe : 독립적인 프로세스간 통신



pipe

- pipefd[2] : 파이프로 사용할 파일 디스크립터 (2 개)
- pipefd[0] : 읽기 , pipefd[1] : 쓰기

```
#include <unistd.h>

int pipe(int pipefd[2]);
```

[함수 원형]

Anonymous pipe

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>


int main(void)
{
    int     fd[2];
    pid_t   pid;
    char*   message = "Hello, World!\n";
    char    buf[128] = {};

    pipe(fd);

    pid = fork();
    if(pid == 0) {
        printf("Child Process => PID: %d, PPID: %d\n", getpid(), getppid());
        write(fd[1], message, (strlen(message)));
        exit(0);
    }
    else if (pid > 0 ) {
               printf("Parent Process => PID: %d, PPID: %d\n", getpid(), getppid());
        read(fd[0], buf, sizeof(buf));
        printf("%s\n", buf);
    }

    return 0;
}
```

mkfifo

- pathname : FIFO 파일을 생성할 경로
- mode : 접근 권한 지정

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

int mkfifo(const char *pathname, mode_t mode);
```

[함수 원형]

Named pipe

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>

int main() {
    int fd;
    char* myfifo = "/tmp/myfifo";
    char *message = "Hello, World!\n";

    mkfifo(myfifo, 0666);

    fd = open(myfifo, O_WRONLY);
    write(fd, message, strlen(message));
    close(fd);

    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>

int main() {
    int fd;
    char* myfifo = "/tmp/myfifo";
    char buf[128];

    mkfifo(myfifo, 0666);

    fd = open(myfifo, O_RDONLY);
    read(fd, buf, 128);
    printf("%s \n", buf);
    close(fd);

    return 0;
}
```

03
socket

socket

- 여러 컴퓨터에 있는 프로세스 간 통신하는 방법
- OS 가 제공하는 네트워크 연결 도구
- 보통 데이터를 보내는 곳을 클라이언트 , 데이터를 받는 곳을 서버라고 칭함

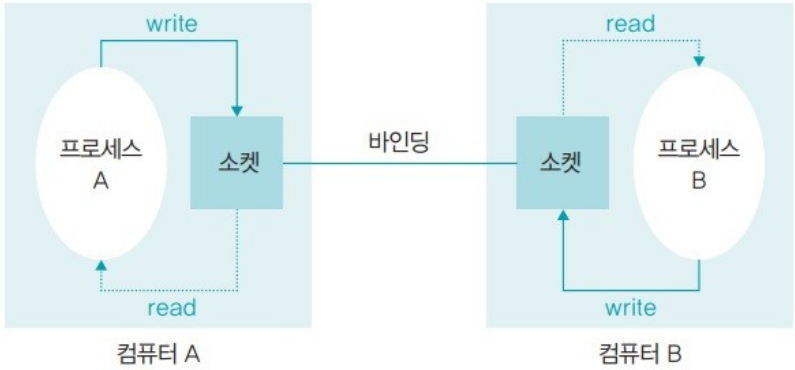


그림 5-9 **소켓을 이용한 통신**

socket

- IP : 네트워크 상에서 컴퓨터를 식별하는 **고유한 주소**
- PORT : 컴퓨터 내에서 실행중인 특정 프로그램을 식별하는 번호
- bind : 소켓에 특정 IP 주소와 PORT 번호를 할당

- 0~1023 번 포트는 주요 통신을 위해 이미 정해진 포트

PORT	프로토콜	설명
20	FTP	파일 전송
22	SSH	원격 로그인
80	HTTP	웹

UDP Socket - Client

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

#define PORT 9005

int main() {
    int sockfd;
    char* hello = "Hello World!!";
    struct sockaddr_in servaddr;

    // Creating socket file descriptor
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd == -1) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    // Filling server information
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(PORT);
    servaddr.sin_addr.s_addr = INADDR_ANY;

    sendto(sockfd, hello, strlen(hello), 0,
            (const struct sockaddr *) &servaddr, sizeof(servaddr));
    close(sockfd);

    return 0;
}
```

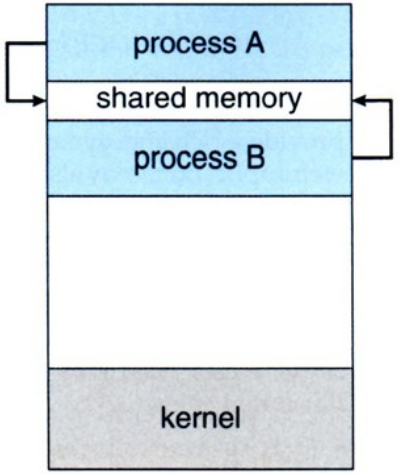
UDP Socket - Server

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

#define PORT 9005


int main() {
    int sockfd;
    char buffer[128];
    char *hello = "Hello from server";
    struct sockaddr_in servaddr, cliaddr;

    sockfd = socket(AF_INET, SOCK_DGRAM, 0);

    servaddr.sin_family      = AF_INET;
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port        = htons(PORT);

    bind(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr));

    int len = sizeof(cliaddr);
    recvfrom(sockfd, buffer, 128, 0, (struct sockaddr *) &cliaddr, &len);
    printf("%s\n", buffer);
    close(sockfd);

    return 0;
}
```

04
Shared Memory

shared memory

- 여러 프로세스가 공동으로 사용하는 메모리
- IPC 기법 중 처리속도가 가장 빠르다



Shared memory.

shared memory

- Key : 세그먼트를 식별하는 값
- size : 공유할 메모리 크기
- shmflg : 공유 메모리에 대한 읽기 / 쓰기 권한

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

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

[함수 원형]

- shmid : shmget 으로 생성한 공유 메모리 식별자
- shmaddr : 공유메모리 연결할 주소
- shmflg : 공유 메모리에 대한 읽기 / 쓰기 권한

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

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

[함수 원형]

shared memory

```
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdlib.h>
#include <stdio.h>

int main(){
        int shmid;
        int *num;
        key_t key=987654;
        void *memory_segment=NULL;

        if((shmid=shmget(key, sizeof(int),IPC_CREAT|0666))== -1){
                printf("shmget failed\n");
                exit(0);
        }

        if((memory_segment=shmat(shmid, NULL, 0)) == (void*)-1){
                printf("shmat failed\n");
                exit(0);
        }

        num=(int*)memory_segment;
        (*num)++;
        printf("shared memory value :%d\n",(*num));
        return 0;
}
```

$ ipcs –m

- IPC 에서 사용하는 고유 메모리 정보

$ ipcrm –m <shmid>

- 특정 고유 메모리 제거