HALLYM UNIVERSITY

CIC Convergence Information & Communications

(Operating System) Practice -7-

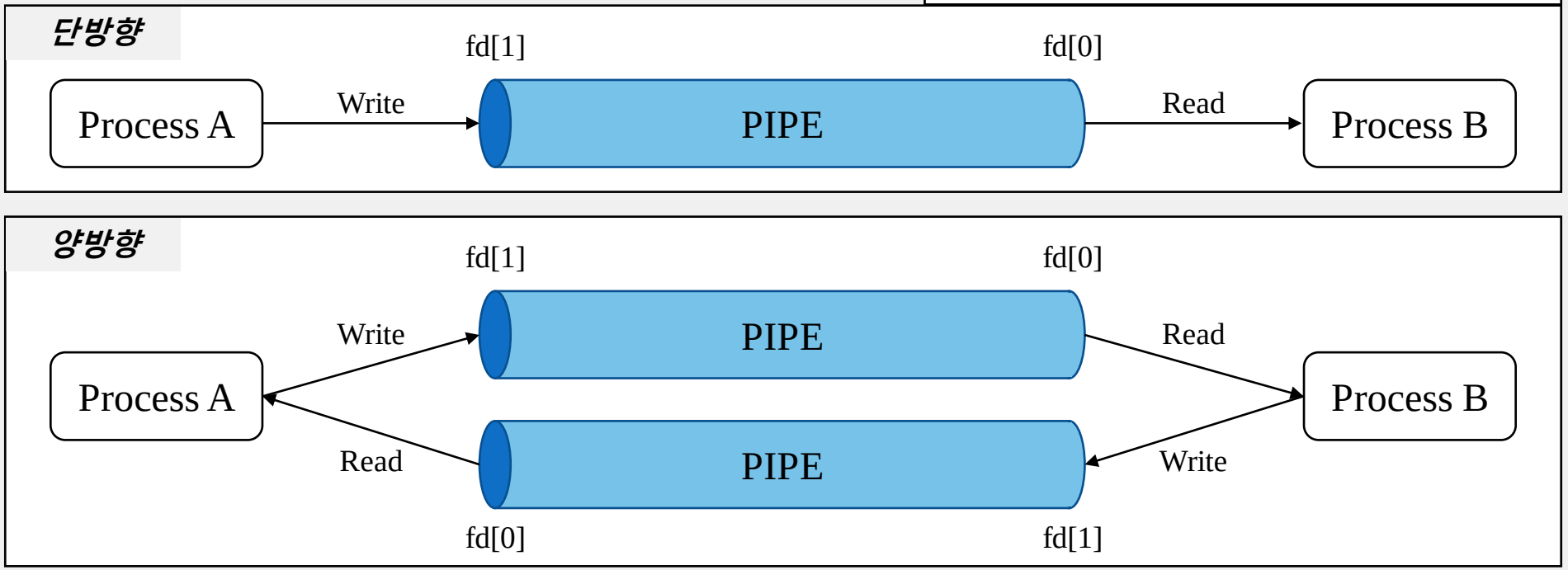
Pipe & Named pipe

Index

Pipe overview

- **Pipe**
  - Related process간 Inter process communication을 위해 사용함
    - ✓ Related process의 예시 → 자식-부모 프로세스
    - ✓ 별도의 ID가 없음 → 즉, 이름이 없음, fd를 통해서만 사용가능.
  - Uni-directional byte stream

사용하지 않는 전송방향은 close 해줘야 함



Exec overview

- **pipe**
  - ✓ 형태
    - pipe(int pipefd[2])
  - ✓ 인수
    - Int pipefd[2]: 생성될 pipefd[2]를 저장할 버퍼
      - » pipefd[0] → reader-side fd
      - » pipefd[1] → writer-side fd
  - ✓ 반환
    - Int: 비정상 작동시 -1을 반환

Pipe practice -1

- **Pipe practice – 1**
  - 단방향 예제

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

#define MAX_BUF 1024
int main(){
    int fd[2];
    pid_t pid;
    char buf[MAX_BUF];

    if(pipe(fd)<0){
        printf("pipe error\n");
        exit(1);
    }
    pid = fork();
    if(pid<0){
        printf("fork error\n");
        exit(1);
    }

    if(pid>0){ // parent process
        close(fd[0]); // close pipe read
        strcpy(buf, "I am your father.");
        write(fd[1], buf, strlen(buf));
    }
    else{ //child process
        close(fd[1]); // close pipe write
        read(fd[0], buf, MAX_BUF);
        printf("Child received the following message from its parent:\n --> %s\n",buf);
    }
    exit(0);
}
```

Pipe practice -1

- **Pipe practice – 2**
  - 양방향 예제

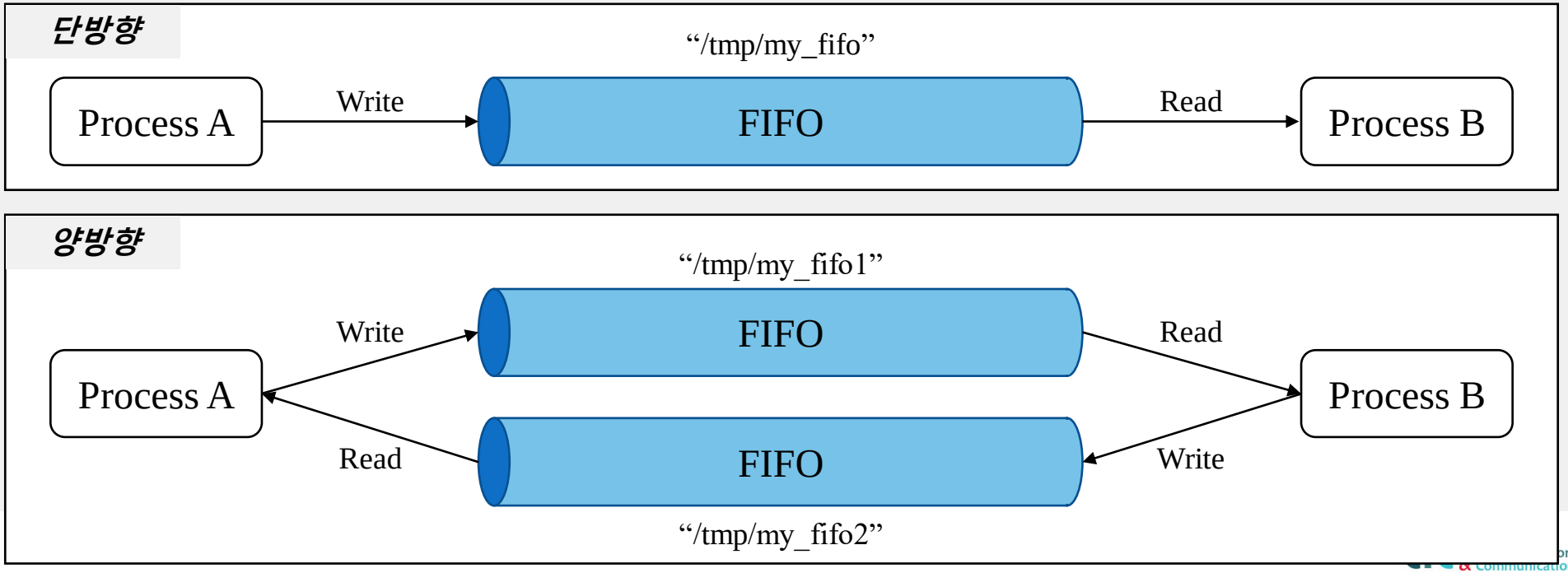
```c
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#define MAX_BUF 1024
int main(){
    int fd_p2c[2], fd_c2p[2];
    pid_t pid;
    char buf[MAX_BUF];

    if(pipe(fd_p2c) < 0){
        printf("pipe error\n");
        exit(1);
    }
    if(pipe(fd_c2p) < 0){
        printf("pipe error\n");
        exit(1);
    }
    pid=fork();
    if(pid<0){
        printf("fork error\n");
        exit(1);
    }
    else if(pid>0){ //parent process
        close(fd_p2c[0]); // p2c pipe read close
        close(fd_c2p[1]); // c2p pipe write close
        for(int i=0;i<10;i++){
            sprintf(buf, "(parent %d)", i);
            write(fd_p2c[1], buf, MAX_BUF);
            memset(buf, 0, sizeof(buf)); // memory initialization
            read(fd_c2p[0], buf, MAX_BUF);
            printf("parent recieved message: %s\n", buf);
            sleep(1);
        }
    }
    else { // child process
        close(fd_c2p[0]); // c2p pipe read close
        close(fd_p2c[1]); // p2c pipe write close
        for(int i=1000;i<1010;i++){
            sprintf(buf, "(child %d)", i);
            write(fd_c2p[1], buf, MAX_BUF);
            memset(buf, 0, sizeof(buf)); // memory initialization
            read(fd_p2c[0], buf, MAX_BUF);
            printf("\tchild recieved message: %s\n", buf);
            sleep(1);
        }
    }
    exit(0);
}
```

Named pipe overview

- **Named pipe (FIFO)**
  - 기본적인 개념은 PIPE와 동일하나, unrelated process간 데이터 통신 가능
  - 파일 경로 = ID
    - ✓ 즉, 프로세스가 파일 경로만 알고 있다면, Named pipe를 통해 데이터 통신 가능.
  - Named pipe 생성과, Open이 분리되어 있음
  - Open() 시 수신 측과 송신 측이 서로 동기화 됨
    - ✓ 즉, 양쪽이 모두 Open()을 수행해야만, 데이터 주고 받기 가능



Named pipe overview

- **Named Pipe (FIFO)**
  - ✓ 형태
    - mkfifo(const char *pathname, mode_t mode)
  - ✓ 인수
    - const char *filename: 생성할 named pipe의 파일 경로
    - Mode: permission
  - ✓ 반환
    - Int: 비정상 작동시 -1을 반환
  - ✓ mkfifo로 named pipe를 생성한 뒤, **open/read/write**를 통해 데이터 통신을 수행

Named pipe practice

- **Named pipe practice**

fifo_sender.c

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

int main(int argc, char* argv[]){
    int arr[5];
    srand(time(NULL));
    int i;
    for (i=0;i<5;i++){
        arr[i] = rand() % 100; // 0~100
    }
    if(mkfifo("myfifo",0777) == -1){ // 0777: All permissions are given.
        printf("fifo error\n");
        exit(-1);
    }
    int fd = open("myfifo", O_WRONLY);
    for (i=0;i<5;i++){
        printf("Writing %d\n",arr[i]);
        if(write(fd, &arr[i], sizeof(int))==-1){
            printf("write error\n");
            exit(-1);
        }
    }
    close(fd);
    exit(0);
}
```

fifo_receiver.c

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

int main(int argc, char* argv[]){
    int fd = open("myfifo", O_RDONLY);
    int arr[5];
    int i;
    for (i=0;i<5;i++){
        if(read(fd, &arr[i], sizeof(int))==-1){
            printf("read error\n");
            exit(-1);
        }
    printf("Reading %d\n",arr[i]);
    }
    close(fd);
    exit(0);
}
```