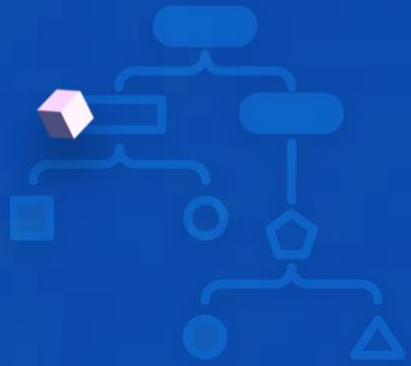

강원혁신플랫폼

# 리눅스프로그래밍

## 파이프(pipe)



Linux 시스템에서 한 쌍의 관련 프로세스 간에 통신을 허용하는 간단한 형태의 IPC는 무엇인가요?

**파이프(pipe)**

# 학습목표



**학습 내용**

1. 파이프와 매개변수
2. 파이프를 이용한 두 프로세스 간의 통신 관계
3. pipe.c, fifo_server.c, fifo_client.c

**학습 목표**

- 파이프와 매개변수에 대해 설명할 수 있다.
- 파이프 이용한 프로세스 간의 통신을 이해할 수 있다.
- Named Pipe(FIFO) 문제에 대해 설명할 수 있다.

# 파이프와 매개변수

# 파이프를 이용한 두 프로세스 간의 통신 관계

## 파이프의 특징

- 한 쌍의 관련 프로세스 간에 프로세스 간 통신(IPC)을 허용하는 단방향 통신 채널
- 한 프로세스에서 다른 프로세스로 데이터를 전달함

```
#include <unistd.h>
int pipe(int pipefd[2]);
```

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

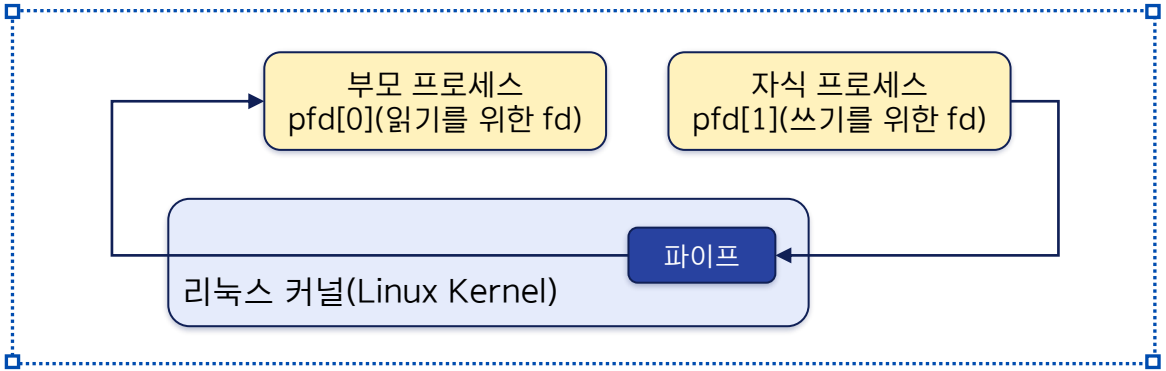
# 파이프를 이용한 두 프로세스 간의 통신 관계

## 매개변수

[ 파이프의 읽기 및 쓰기 끝에 대한
파일 디스크립터가 저장될 두 개의 정수 배열 ]

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

# 파이프를 이용한 두 프로세스 간의 통신 관계




pipefd[0]

[ 읽기 끝 ]

pipefd[1]

[ 쓰기 끝 ]

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

# 따라하기 pipe.c



```
#include <stdio.h>              /* printf( ) 함수를 위해 사용 */
#include <unistd.h>
#include <sys/wait.h>           /* waitpid() 함수를 위해 사용 */

int main(int argc, char **argv)
{
    pid_t pid;
    int pfd[2];
    char line[BUFSIZ];          /* <stdio.h> 파일에 정의된 버퍼 크기로 설정 */
    int status;
```

# 따라하기 pipe.c



```
if(pipe(pfd) < 0) {        /* pipe( ) 함수를 이용해서 파이프 생성 */
    perror("pipe()");
    return -1;
}
```

# 따라하기 pipe.c



```
if((pid = fork()) < 0) {    /* fork( ) 함수를 이용해서 프로세스 생성 */
    perror("fork()");
    return -1;
} else if(pid == 0) {       /* 자식 프로세스인 경우의 처리 */
    close(pfd[0]);          /* 읽기를 위한 파일 디스크립터 닫기 */
    dup2(pfd[1], 1);        /* 쓰기를 위한 파일 디스크립터를 표준 출력(1)으로 변경 */
    execl("/bin/date", "date", NULL);              /* date 명령어 수행 */
    close(pfd[1]);          /* 쓰기를 위한 파일 디스크립터 닫기 */
    _exit(127);
} else {                    /* 부모 프로세스인 경우의 처리 */
```

# 따라하기 pipe.c



```
    } else {                    /* 부모 프로세스인 경우의 처리 */
        close(pfd[1]);          /* 쓰기를 위한 파일 디스크립터 닫기 */

        if(read(pfd[0], line, BUFSIZ) < 0) {    /* 파일 디스크립터로부터 데이터 읽기 */
            perror("read()");
            return -1;
        }
        printf("%s", line);     /* 파일 디스크립터로부터 읽은 내용을 화면에 표시 */
        close(pfd[0]);          /* 읽기를 위한 파일 디스크립터 닫기 */
        waitpid(pid, &status, 0);               /* 자식 프로세스의 종료를 기다리기 */
    }
    return 0;
}
```

# 따라하기 pipe.c



## 실행결과

- 자식 ps에서 넘겨 받은 execl( " /bin/date " , " date", NULL); 의 결과를 부모 ps가 화면에 출력함

```
$ gcc -o pipe pipe.c
$
$ ./pipe
Sat Jul  8 19:31:43 KST 2023
```

# 따라하기 fifo_server.c



```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>                    /* read(), close(), unlink() 등의 시스템 콜을 위한
헤더 파일 */
#include <sys/stat.h>
#define FIFOFILE "fifo"

int main(int argc, char **argv) {
    int n, fd;
    char buf[BUFSIZ];
    unlink(FIFOFILE);                  /* 기존의 FIFO 파일을 삭제한다. */
```

# 따라하기 fifo_server.c



```
    if(mkfifo(FIFOFILE, 0666) < 0) {    /* 새로운 FIFO 파일을 생성한다. */
        perror("mkfifo()");        return -1;
    }
    if((fd = open(FIFOFILE, O_RDONLY)) < 0) {          /* FIFO를 연다. */
        perror("open()");        return -1;
    }
    while((n = read(fd, buf, sizeof(buf))) > 0)        /* FIFO로부터 데이터를 받아온다. */
        printf("%s", buf);              /* 읽어온 데이터를 화면에 출력한다. */

    close(fd);
    return 0;
}
```

# 따라하기 fifo_client.c



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

#define FIFOFILE "fifo"

int main(int argc, char **argv)
{
    int n, fd;
    char buf[BUFSIZ];
```

# 따라하기 fifo_client.c



```
    if((fd = open(FIFOFILE, O_WRONLY)) < 0) {          /* FIFO를 연다. */
        perror("open()");
        return -1;
    }

    while ((n = read(0, buf, sizeof(buf))) > 0)        /* 키보드로부터 데이터를 입력받는다 */
        write(fd, buf, n);                             /* FIFO로 데이터를 보낸다. */

    close(fd);

    return 0;
}
```

# 따라하기 fifo 실행결과



- 실행결과
  - fifo_client는 키보드 입력을 받아, FIFOFILE에 출력
  - fifo_server는 FIFOFILE 생성, FIFOFILE의 내용을 읽어 화면에 출력

```
$ gcc -o fifo_server fifo_server.c
$ gcc -o fifo_client fifo_client.c
$
$ ./fifo_server &
[1] 3499
$ ./fifo_client
hello world
hello world
^C
```

# 따라하기 fifo 실행결과



## 실행결과

```
[1]+  Done                    ./fifo_server
$
$ ls -al fifo
prw-r--r-- 1 freetime freetime 0 Jul  8 19:39 fifo
$
```

# 요약정리



01 • 파이프와 매개변수

02 • 파이프를 이용한 두 프로세스 간의 통신 관계

03 • Named Pipe(FIFO)