Pipe

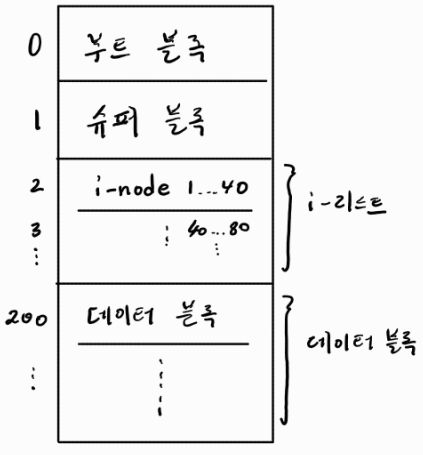
. 2 프로세스 간의 연결

. 한 프로세스의 출력이 다른 프로세스의 표준 입력이 됨) ⟶ Inter-Process Communication

특징
- 단방향 통신
  - 하나의 프로세스는 파이프에 write
  - 다른 하나의 프로세스는 파이프에서 read] ⟶ 파이프 = buffer (virtual file / main memory)
- 생성프로세스와 모든 하위 프로세스가 공유 가능
- 하나는 write, 하나는 read ⟶ Read/Write Sync 문제 존재
  - ⇒ Write 전 (비어있는 상태) 읽기를 시도 ⟶ 기록 완료까지 대기
- 프로세스의 Open file table에서 사용가능한 첫 두개의 키로 찾고 이를 파이프의 읽기끝/쓰기 끝으로 할당

예) Open file table

파일시스템



부트 블록 : 리눅스 시작시 사용되는 부트스트랩 코드

슈퍼 블록 : 전체 파일 시스템에 대한 정보 저장
(총 블록 수/사용가능 i-node 개수/ ---)

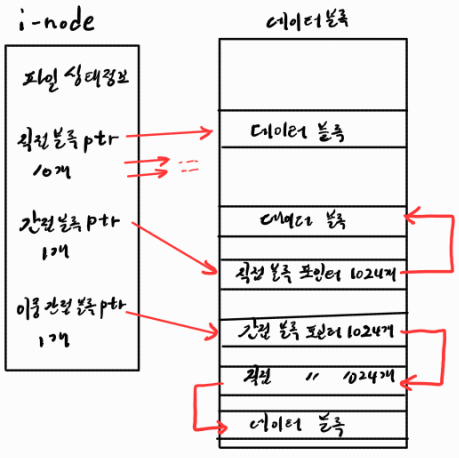
i-node 리스트 : 각 파일을 나타내는 모든 i-node들의 리스트
한 블록당 약 40개의 i-node

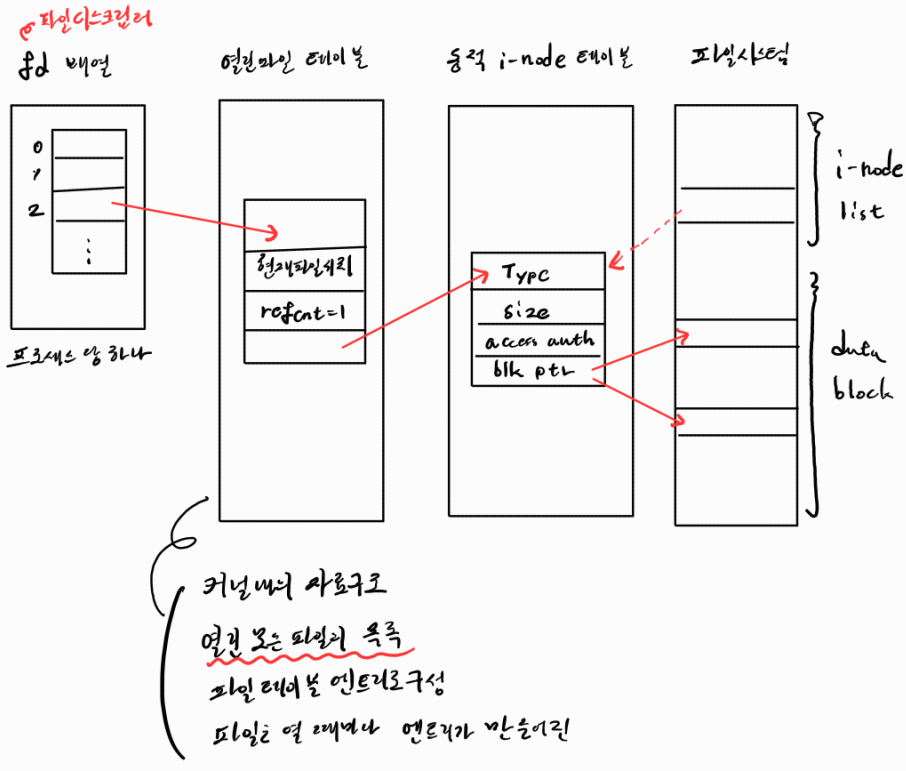
데이터 블록 : 파일의 내용(데이터)을 저장하기 위한 블록

파일 입출력

```
fd = open("file", O_RDONLY);
```

i-node [한 파일은 하나의 i-node를 가짐
파일의 상태정보 (타입 / 크기 / 퍼미션 / 소유자 / ---)

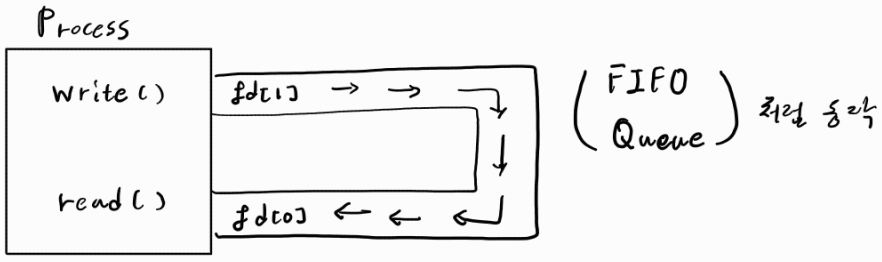




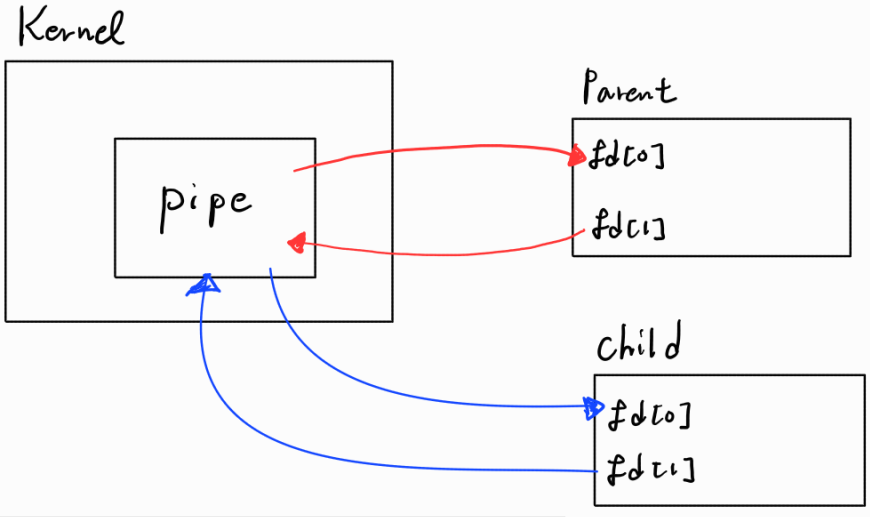
int pipe(int fd[2]);

fd[0] : read end

fd[1] : write end



How to IPC? → fork




```c
#include <stdio.h>
#include <unistd.h>
#define MSGSIZE 16
char* msg1 = "hello, world #1";
char* msg2 = "hello, world #2";
char* msg3 = "hello, world #3";

int main()
{
    char inbuf[MSGSIZE];
    int p[2], pid, nbytes;

    if (pipe(p) < 0)
        exit(1);

    /* continued */
    if ((pid = fork()) > 0) {
        write(p[1], msg1, MSGSIZE);
        write(p[1], msg2, MSGSIZE);
        write(p[1], msg3, MSGSIZE);

        // Adding this line will
        // not hang the program
        // close(p[1]);
        wait(NULL);
    }

    else {
        // Adding this line will
        // not hang the program
        // close(p[1]);
        while ((nbytes = read(p[0], inbuf, MSGSIZE)) > 0)
            printf("% s\n", inbuf);
        if (nbytes != 0)
            exit(2);
        printf("Finished reading\n");
    }
    return 0;
}
```

```
hello world, #1
hello world, #2
hello world, #3
(hangs)         //program does not terminate but hangs
```

https://www.geeksforgeeks.org/pipe-system-call/
https://www.geeksforgeeks.org/c-program-demonstrate-fork-and-pipe/

pipe system call (based on linux 5.8)

- include/linux/syscalls.h

```
asmlinkage long sys_pipe2(int __user *fildes, int flags);
asmlinkage long sys_pipe(int __user *fildes);
```

- arch/x86/entry/syscalls/syscall_64.tbl

```
22   common   pipe    sys_pipe
293  common   pipe2   sys_pipe2
```

- include/uapi/asm-generic/unistd.h

```
#define __NR_pipe2 59
__SYSCALL(__NR_pipe2, sys_pipe2)
```

- fs/pipe.c & include/linux/pipe_fs_i.h