IPC (Inter Process Communication)

한양대학교 소프트웨어학부

Dept. of Division of Computer Science

Hanyang University

CSE4009
System Programming

HANYANG UNIVERSITY

PIPE

- 운영체제가 제공하는 프로세스간 통신 채널로서 특별한 타입의 파일
  - 일반 파일과 달리 메모리에 저장되지 않고 운영체제가 관리하는 임시파일
  - 데이터 저장용이 아닌 프로세스간 데이터 전달용으로 사용

- 파이프를 이용한 프로세스간 통신
  - 송신측은 파이프에 데이터를 쓰고 수신측은 파이프에서 데이터를 읽음
  - 스트림 채널을 제공
  - 송신된 데이터는 바이트 순서 유지

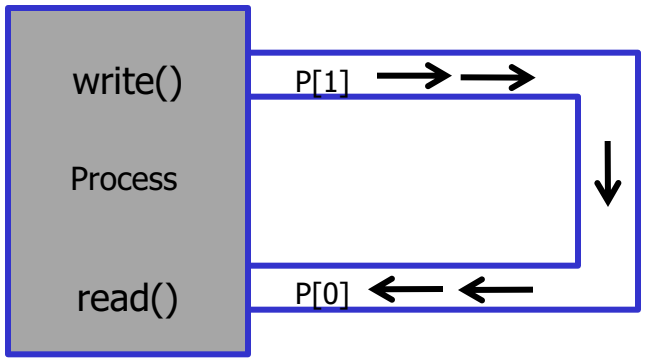
IPC – pipe

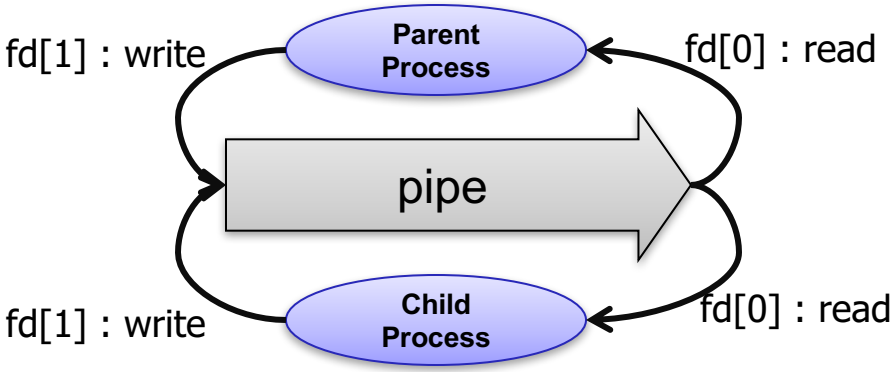
사용법

```
#include <unistd.h>

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

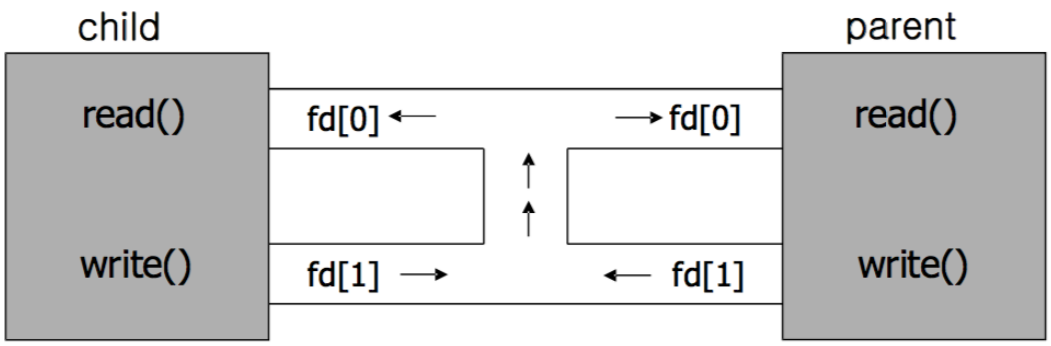
파이프를 열기 위해서는 pipe() 함수 를 사용
- 하나의 파이프를 생성하면 두개 (읽기, 쓰기)의 파일 디스크립터가 생성됨
- fd[0]은 읽기 용, fd[1]은 쓰기 용
- 파이프를 생성한 프로세스가 fork()를 호출하면 자식 프로세스는 부모 프로세스의 파이프를 공유



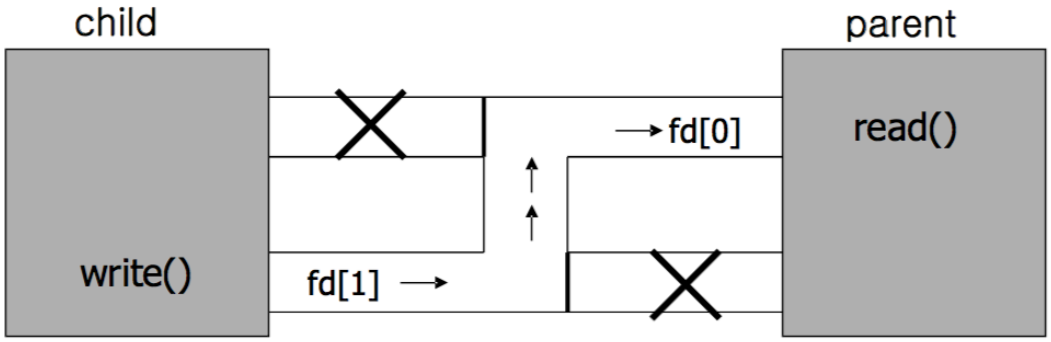


IPC – pipe (cont'd)

- 파이프를 이용한 부모와 자식 프로세스간 통신



Fig(a). Unnamed pipe: before closing unnecessary file descriptor



Fig(b). unnamed pipe: after closing unnecessary file descriptor

PIPE의 크기

- 파이프의 크기와 프로세스 제어
  - 파이프의 크기는 유한하다
  - POSIX에는 최소 512 바이트로 정의되어 있다
  - 파이프가 full인데 write하면write하는 프로세스의 수행이 잠시 중단된다
  - 파이프가 empty인데 read하면 read하는 프로세스가 block된다

- 파이프 닫기
  - 파이프에 write하는 모든 프로세스가 이 파이프를 닫으면 이 파이프를 read 하는 프로세스의 read 시스템 호출은 0을 반환한다
  - 파이프에 read 하는 모든 프로세스가 이 파이프를 닫으면 이 파이프에 write하는 프로세스는 신호 SIGPIPE를 받는다

Example – pipe #1

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

#define MSGSIZE 16

char *msg1 = "Hello, World #1";
char *msg2 = "Hello, World #2";
char *msg3 = "Hello, World #3";

int main(void)
{
        char buf[MSGSIZE];
        int p[2], i;

        /* Pipe Open */
        if (pipe(p) == -1) {
                perror("pipe call");
                exit(1);
        }
```

```c
/* Pipe Write */
write(p[1], msg1, MSGSIZE);
write(p[1], msg2, MSGSIZE);
write(p[1], msg3, MSGSIZE);

/* Pipe Read */
for (i = 0; i < 3; i++) {
        read(p[0], buf, MSGSIZE);
        printf("%s\n", buf);
}

return 0;
```

```
[TA3@localhost lab10]$ ./ex1
Hello, World #1
Hello, World #2
Hello, World #3
[TA3@localhost lab10]$
```

Example – pipe #2

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

#define MSGSIZE 16
char *msg1 = "Hello, World #1";
char *msg2 = "Hello, World #2";
char *msg3 = "Hello, World #3";

int main(void)
{
        char buf[MSGSIZE];
        int p[2], pid, i;

        /* Pipe Open */
        if (pipe(p) == -1) {
                perror("pipe call");
                exit(1);
        }
        pid = fork();
        if (pid < 0) {
                perror("fork call");
                exit(1);
        } else if (pid == 0) {
                /* 자식일 경우 읽기 파일 기술자를 닫고, 파이프에 쓴다. */
                close(p[0]);
                write(p[1], msg1, MSGSIZE);
                write(p[1], msg2, MSGSIZE);
                write(p[1], msg3, MSGSIZE);
                exit(1);
        } else {
                /* 부모일 경우 쓰기 파일 기술자를 닫고, 파이프로 부터 읽는다. */
                close(p[1]);
                for (i = 0; i < 3; i++) {
                        read(p[0], buf, MSGSIZE);
                        printf("%s\n", buf);
                }
        }
        return 0;
}
```

```
[TA3@localhost lab10]$ ./ex2
Hello, World #1
Hello, World #2
Hello, World #3
[TA3@localhost lab10]$
```

HANYANG UNIVERSITY

Example – pipe #3 (unblocked read,write) (1)

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>

#define MSGSIZE 6

int parent(int*);
int child(int*);

char *msg1 = "Hello";
char *msg2 = "Bye!!";

int fatal(char *s)
{
        perror(s);
        exit(1);
}

int main(void)
{
        int p[2];

        if (pipe(p) == -1)
                fatal("pipe call");

        if (fcntl(p[0], F_SETFL, O_NONBLOCK) == -1)
                fatal("fcntl call");

        switch (fork()) {
                case -1:
                        fatal("fork call");
                case 0:
                        child(p);
                default:
                        parent(p);
        }
}
```

```c
int parent(int p[2])
{
        int nread;
        char buf[MSGSIZE];

        close(p[1]);
        for (;;) {
                switch (nread = read(p[0], buf, MSGSIZE)) {
                        case -1: /* 파이프 검사 */
                                if (errno == EAGAIN) {
                                        printf("(pipe empty)\n");
                                        sleep(1);
                                        break;
                                } else
                                        fatal("read call");
                        case 0:
                                /* 파이프가 닫힘 */
                                printf("End of Conversation\n");
                                exit(1);
                        default:
                                printf("MSG=%s\n", buf);
                }
        }
}

int child(int p[2])
{
        int count;
        close(p[0]);
        for (count = 0; count < 3; count++) {
                write(p[1], msg1, MSGSIZE);
                sleep(3);
        }

        /* 마지막 메시지 */
        write(p[1], msg2, MSGSIZE);
        exit(1);
}
```

Example – pipe #3 (unblocked read,write) (2)

```
[TA3@localhost lab10]$ ./ex3
(pipe empty)
MSG=Hello
(pipe empty)
(pipe empty)
MSG=Hello
(pipe empty)
(pipe empty)
(pipe empty)
MSG=Hello
(pipe empty)
(pipe empty)
(pipe empty)
MSG=Bye!!
End of Conversation
[TA3@localhost lab10]$
```

Example – pipe #4 pipe 구현

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>

int fatal(char *s)
{
        perror(s);
        exit(1);
}

int join(char *com1[], char *com2[])
{
        int p[2], status;

        switch(fork()) {
                case -1:
                        fatal("1st fork call in join");
                case 0:
                        break;
                default:
                        wait(&status);
                        return status;
        }
        if (pipe(p) == -1)
                fatal("pipe call in join");
                switch (fork()) {
                        case -1:
                                fatal("2nd fork call in join");
                        case 0:
                                // write process
                                dup2(p[1], 1); // == close(1); dup(p[1]);
                                close(p[0]);
                                close(p[1]);
                                execvp(com1[0], com1);
                                fatal("1st execvp call in join");
                        default:
                                // read process
                                dup2(p[0], 0); // == close(0); dup(p[0]);
                                close(p[0]);
                                close(p[1]);
                                execvp(com2[0], com2);
                                fatal("2nd execvp call in join");
                }
}

int main(void)
{
        char *one[4] = {"ls", "-l", "/usr/lib", NULL};
        char *two[3] = {"grep", "^d", NULL};
        int ret;

        ret = join(one, two);
        printf("join returned %d\n", ret);
        return 0;
}
```

IPC – FIFO

사용법

```
#include <sys/types.h>
#include <sys/stat.h>
int mkfifo(const chat *pathname, mode_t mode);
```

- 파이프는 fork()로 만들어진 프로세스들 사이의 통신에만 사용 가능한 제약이 있음
  - 제한을 극복하기 위해 파이프에 이름을 지정하고 임의의 다른 프로세스에서 파이프에 접근하도록 한 것을 named pipe 또는 FIFO라 함
  - pathname : 파이프의 이름, 경로명이 없으면 현재 디렉토리
  - mode : FIFO의 파일 접근 권한 설정

-읽기/쓰기 수행

  - FIFO를 생성한 후 FIFO를 open()해야 함
  - FIFO도 파이프의 일종으로 프로세스간 스트림 채널을 제공

Example – fifo #sender

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <string.h>

#define MSGSIZE 80

char *fifo = "fifo";

int main(int argc, char* argv[])
{
        int fd, i, nwrite;
        char msgbuf[MSGSIZE];
        if (argc < 2) {
                fprintf(stderr, "Usage: ./sender msg... \n");
                exit(1);
        }

        if ((fd = open(fifo, O_WRONLY | O_NONBLOCK)) < 0)
                perror("fifo open failed");

        for (i = 1; i < argc; i++) {
                if (strlen(argv[i]) > MSGSIZE) {
                        fprintf(stderr, "message is too long: %s\n", argv[i]);
                        continue;
                }
                strcpy(msgbuf, argv[i]);
                if ((nwrite = write(fd, msgbuf, MSGSIZE)) == -1)
                        perror("message write failed");
        }
        return 0;
}
```

Example – fifo #receiver

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>

#define MSGSIZE 80

char *fifo = "fifo";

int main(int argc, char* argv[])
{
        int fd;
        char msgbuf[MSGSIZE];
        if (mkfifo(fifo, 0666) == -1) {
                if (errno != EEXIST)
                        perror("receiver: mkfifo");
        }
        if ((fd = open(fifo, O_RDWR)) < 0)
                perror("fifo open failed");

        for (;;) {
                if (read(fd, msgbuf, MSGSIZE) < 0)
                        perror("message read failed");

                printf("message received: %s\n", msgbuf);
        }
}
```

Q & A

- Thank you :)