IPC (Inter Process Communication)

한양대학교 컴퓨터공학과

Dept. of Computer Science & Engineering
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 순서가 유지

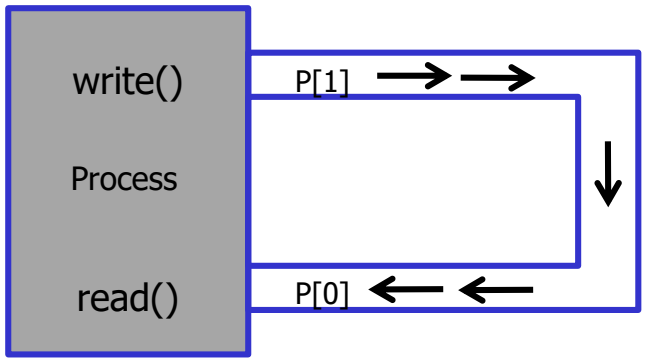
IPC – pipe

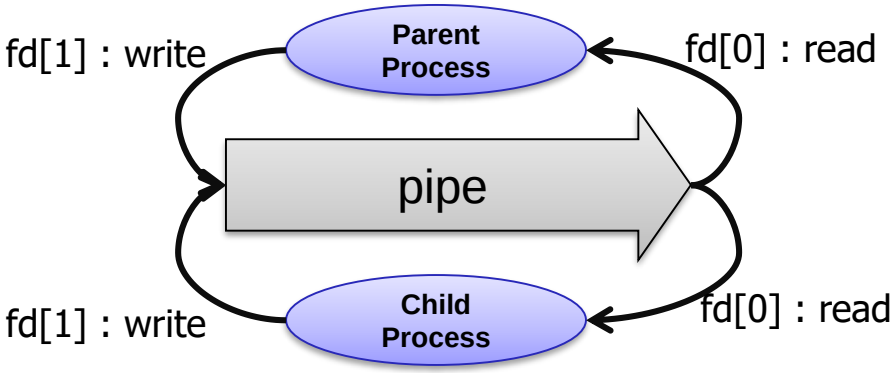
사용법

```
#include <unistd.h>

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

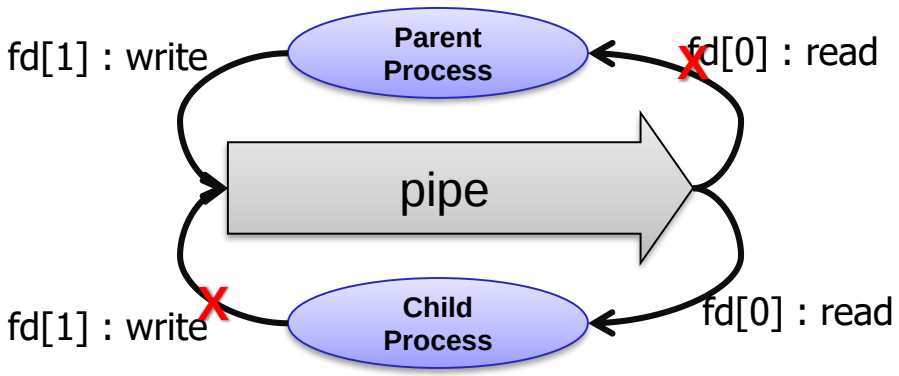
파이프를 열기 위해서는 pipe() 함수를 사용
- 하나의 파이프를 생성하면 두개 (읽기, 쓰기)의 파일 디스크립터가 생성됨
- fd[0]은 읽기 용, fd[1]은 쓰기 용
- 파이프를 생성한 프로세스가 fork()를 호출하면 자식 프로세스는 부모 프로세스의 파이프를 공유



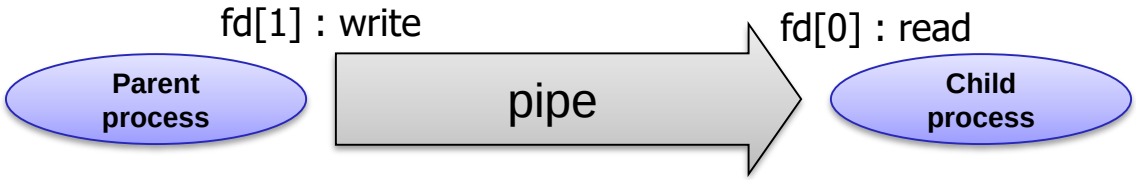


IPC – pipe (cont'd)

- 사용하지 않는 파일 디스크립터를 닫은 상태



- 파이프를 이용한 부모와 자식 프로세스간 통신



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
/* 파이프의 첫 번째 예*/
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

/* 아래 숫자는 마지막에 null 문자들을 포함*/
#define MSGSIZE     16
char *msg1 = "hello, world #1";
char *msg2 = "hello, world #2";
char *msg3 = "hello, world #3";
main()
{
  char inbuf[MSGSIZE];
  int p[2], j;

  /* 파이프를 개방한다*/
  if(pipe(p) == -1)
  {
    perror("pipecall");
    exit(1);
  }
  /* 파이프에 쓴다*/
  write(p[1], msg1, MSGSIZE);
  write(p[1], msg2, MSGSIZE);
  write(p[1], msg3, MSGSIZE);

  /* 파이프로부터 읽는다*/
  for(j= 0; j < 3; j++)
  {
    read (p[0], inbuf, MSGSIZE);
    printf("%s\n", inbuf);
  }
  exit (0);
}
```

Example – pipe #2

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

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

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

  /* 파이프를 개방한다. */
  if (pipe(p) == -1)
  {
    perror("pipe call");
    exit(1);
  }

  switch (fork()){
    case -1:
      perror("fork call");
      exit(2);

    case 0:
      /* 자식이면 읽기 파일 기술자를 닫고, 파이프에 쓴다*/
      close(p[0]);
      write(p[1], msg1, MSGSIZE);
      write(p[1], msg2, MSGSIZE);
      write(p[1], msg3, MSGSIZE);
      break;

    default:
      /* 만일 부모이면 쓰기 파일 기술자를 닫고,
         파이프로부터 읽는다. */
      close(p[1]);
      for (j = 0; j < 3; j++)
      {
        read(p[0], inbuf, MSGSIZE);
        printf("%s\n", inbuf);
      }

      wait(NULL);
  }

  exit (0);
}
```

Example – pipe #3 (unblocked read,write)

```c
/* O_NONBLOCK 예*/

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

#define MSGSIZE 6

int parent(int*);
int child(int*);
int fatal(char*);

char *msg1 = "hello";
char *msg2 = "bye!!";

main()
{
  int pfd[2];

  /* 파이프를개방한다. */
  if(pipe(pfd) == -1)
    fatal("pipe call");

  /* p[0]의O_NONBLOCK
     플래그를 1로 설정한다. */
  if(fcntl(pfd[0], F_SETFL, O_NONBLOCK) == -1)
    fatal("fcntlcall");

  switch (fork()){
    case -1:
      fatal("forkcall");

    case 0:
      child(pfd);

    default:
      parent(pfd);
  }
}

int parent(int *p)
{
  int nread;
  char buf[MSGSIZE];

  close(p[1]);

  for(;;){
     switch (nread = read(p[0], buf, MSGSIZE)){

       case -1:
       /* 파이프에 아무것도 없는지 검사 */
         if(errno == EAGAIN){
          printf("(pipe empty)\n");
          sleep(1);

          break;
```

Example – pipe #3 (unblocked read,write)

```
        }

        else
         fatal ("read call");

      case 0:
        /* 파이프가 닫혔음 */
        printf("End of conversation\n");
        exit(0);

      default:
        printf("MSG = %s\n", buf);

    }
  }
}

int child(int *p)
{
  int count;
  close(p[0]);

  for (count= 0; count < 3; count++)
  {
    write(p[1], msg1, MSGSIZE);
    sleep(3);
  }
  /* 마지막 메시지를 보낸다. */
  write(p[1], msg2, MSGSIZE);
  exit(0);
}

int fatal(char *s)
{
  /* 오류 메시지를 프린트하고 죽는다. */
  perror(s);
  exit(1);
}
```

```
(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
```

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 다름 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 #1 - 1

```c
/* message sender*/

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

#define MSGSIZ  80

char *fifo= "fifo";

main(int argc, char **argv)
{
  int fd, j, nwrite;
  char msgbuf[MSGSIZ];

  if(argc<2){
    /* argument가 들어오지 않을 경우 */
    fprintf(stderr,"Usage:sendmessage msg...\n");

    /* 사용 방법 출력후 종료 */
    exit(1);
  }

  if((fd=open(fifo, O_WRONLY| O_NONBLOCK)) < 0)
    /* fifo파일을 Non-bloking 모드로 open */
    perror("fifo open failed");
```

Example – fifo #1 - 2

```
  for(j=1; j<argc; j++) {
    /* argument의 개수만큼 loop */
    if(strlen(argv[j]) > MSGSIZ){
      /* 해당 메세지가 MSGSIZ보다 클 경우 */
      fprintf(stderr,"message too long %s\n",argv[j]);
      /* 에러 메시지 출력 후 다음 argument 값으로 넘어감 */
      continue;
    }

    /* 해당 메세지를 msgbuf에 저장한 후 */
    strcpy(msgbuf, argv[j]);

    /* 메시지 전송 */
    if((nwrite= write(fd, msgbuf, MSGSIZ)) == -1)
      perror("message write failed");
  }

  exit(0);
}
```

Example – fifo #2

```c
/* receive message -FIFO를통해메시지를받는다.*/

#include <fcntl.h>
#include <stdio.h>
#include <errno.h>

#define MSGSIZ    80

char *fifo="fifo";

main(int argc, char **argv)
{
  int fd;
  char msgbuf[MSGSIZ];

  /* fifo 생성 */
  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, MSGSIZ) < 0)
      perror("message read failed");

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

```
[2017107775@mango pipe]$ ./fifo2 &
[1] 26542
[2017107775@mango pipe]$ ./fifo1 hello bye
message received:hello
message received:bye
```

실습

- fifo, signal 실습 코드를 이용하여, signal을 받는 경우 해당 시그널에 대한 메시지를 fifo를 통해 fifo2에서 출력해주는 프로그램 작성
  - fifo_sigint.c
  - fifo_sigusr.c
  - fifo2.c (fifo example #2 변경x)

```
[2017107775@mango pipe]$ ./fifo2 &
[1] 15614
[2017107775@mango pipe]$ ./fifo_sigint &
[2] 15615
[2017107775@mango pipe]$ ./fifo_sigusr &
[3] 15616
[2017107775@mango pipe]$ kill -SIGINT 15615
message received:PID: 15615 SIGINT occurrence!!
[2017107775@mango pipe]$ kill -SIGINT 15615
message received:PID: 15615 SIGINT occurrence!!
[2017107775@mango pipe]$ kill -SIGUSR1 15616
message received:15616 SIGUSR1 occurrence!!
[2017107775@mango pipe]$ kill -SIGUSR1 15616
message received:15616 SIGUSR1 occurrence!!
```

Q & A

- Thank you :)