11주차 2차시 이름 없는 파이프

【학습목표】

1. 이름 없는 파이프를 설명할 수 있다.
2. 이름 없는 파이프 관련 함수를 사용할 수 있다.

학습내용1 : 이름 없는 파이프

1. 파이프의 개념

- 두 프로세스간에 통신할 수 있도록 해주는 특수 파일
- 그냥 파이프라고 하면 일반적으로 이름 없는 파이프를 의미
- 이름 없는 파이프는 부모-자식 프로세스 간에 통신할 수 있도록 해줌
- 파이프는 기본적으로 단방향

2. 파이프 생성: popen(3)

- popne 함수는 fork()함수를 실행해 자식 프로세스를 만들고 command에서 지정한 명령을 exec()함수를 실행해 자식 프로세스가 수행하도록 함.
- 성공 시 : 파일 포인터
- 실패 시 : NULL 포인터

```
#include <stdio.h>
FILE *popen(const char *command, const char *mode);
```

- command : 쉘 명령
- mode : “r”(읽기전용 파이프) 또는 “w”(쓰기전용 파이프)

3. 파이프 닫기: pclose(3)

- 지정한 파이프 닫음.
- 관련된 waitpid() 함수 수행 자식 프로세스들이 기다렸다가 리턴
- 성공 시 : 자식 프로세스의 상태 값 리턴
- 실패 시 : -1 리턴

```
#include <stdio.h>
int pclose(FILE *stream);
```

4. pipe(2)

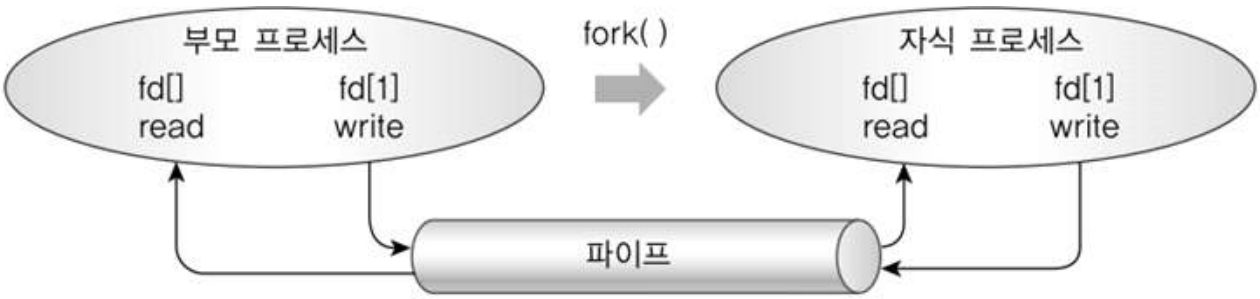
① 기본 개념
- 파이프 만들기: pipe(2)
- 파이프로 사용할 파일기술자 2개를 인자로 지정

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

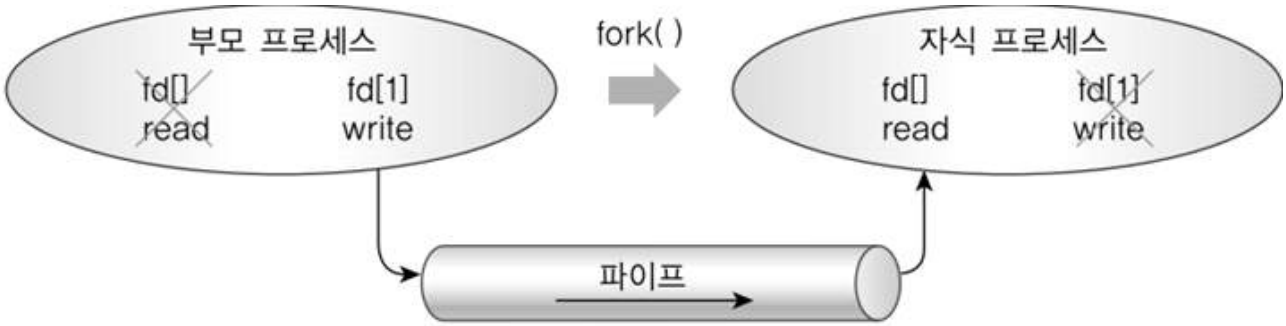
fildes[0]는 읽기, fildes[1]은 쓰기용 파일 기술자

② pipe 함수로 통신과정
첫번째로 pipe() 함수를 호출하여 파이프로 사용할 파일기술자 생성

두번째로 fork 함수로 자식 프로세스 생성. pipe도 자식 프로세스로 복사됨



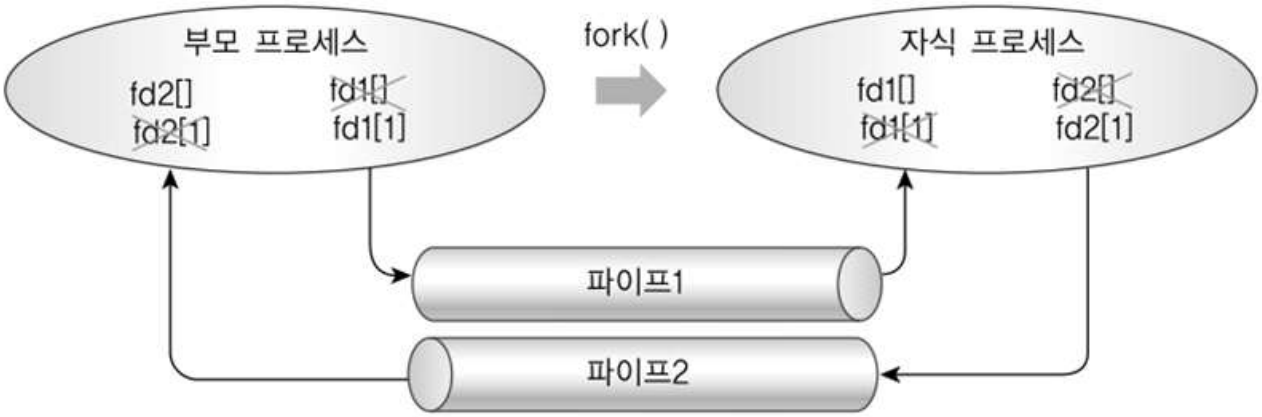
세번째로 통신방향 결정(파이프는 기본적으로 단방향)



5. 양방향 파이프의 활용

* 양방향 파이프의 활용

- 파이프는 기본적으로 단방향이므로 양방향 통신을 위해서는 파이프를 2개 생성한다.



학습내용2 : 이름 없는 파이프 관련 함수

1. popen 함수 사용하기(쓰기 전용)

```
...
04  int main(void) {
05      FILE *fp;
06      int a;
07
08      fp = popen("wc -l", "w");
09      if (fp == NULL) {
10          fprintf(stderr, "popen failed\n");
11          exit(1);
12      }
13
14      for (a = 0; a < 100; a++)
15          fprintf(fp, "test line\n");
16
17      pclose(fp);
18
19      return 0;
20  }
```

"w"모드로 파이프 생성
자식프로세스는 wc –l
명령 수행

자식 프로세스로 출력

2. popen 함수 사용하기(읽기 전용)

```
...
04  int main(void) {
05      FILE *fp;
06      char buf[256];
07
08      fp = popen("date", "r");
09      if (fp == NULL) {
10          fprintf(stderr, "popen failed\n");
11          exit(1);
12      }
13
14      if (fgets(buf, sizeof(buf), fp) == NULL) {
15          fprintf(stderr, "No data from pipe!\n");
16          exit(1);
17      }
18
19      printf("line : %s\n", buf);
20      pclose(fp);
21
22      return 0;
23  }
```

자식 프로세스는
date 명령 실행

읽기모드로 파이프생성

파이프에서 데이터 읽기

```
# ex9_2.out
line : 2010년 2월 5일 금요일 오후 11시 20분 40초
```

3. pipe 함수 사용하기

* pipe() 함수에 의해 생성되는 파일 디스크립터에는 일반 파일처럼 읽기/쓰기 작업이 가능
- 파이프로 사용할 파일기술자 2개를 인자로 지정
- fildes[0]는 읽기, fildes[1]은 쓰기용 파일 기술자

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

- 성공 시 : 0 을 반환실패 시 : -1 반환

```
...
06  int main(void) {
07      int fd[2];
08      pid_t pid;
09      char buf[257];
10 int len, status;
11
12      if (pipe(fd) == -1) {
13          perror("pipe");
14          exit(1);
15      }
16
17      switch (pid = fork()) {
18          case -1 :
19              perror("fork");
20              exit(1);
21              break;
22          case 0 : /* child */
23              close(fd[1]);
24              write(1, "Child Process:", 15);
25              len = read(fd[0], buf, 256);
26              write(1, buf, len);
27              close(fd[0]);
28              break;
29          default :
30              close(fd[0]);
31              buf[0] = '\0';
32              write(fd[1], "Test Message\n", 14);
33              close(fd[1]);
34              waitpid(pid, &status, 0);
35              break;
36      }
37
38      return 0;
39  }
```

파이프 생성

fork로 자식 프로세스 생성

자식 프로세스는 파이프에서 읽을 것이므로 쓰기용 파일 기술자(fd[1])를 닫는다.

파이프에서 읽기

부모 프로세스는 파이프에 쓸 것이므로 읽기용 파일기술자(fd[0])를 닫는다.

파이프에 텍스트를 출력

```
# ex9_3.out
Child Process:Test Message
```

4. pipe 함수 사용하기(2)

* 양방향 통신하기

```
...
07  int main(void) {
08      int fd1[2], fd2[2];
09      pid_t pid;
10      char buf[257];
11      int len, status;
12
13      if (pipe(fd1) == -1) {
14          perror("pipe");
15          exit(1);
16      }
17
18      if (pipe(fd2) == -1) {
19          perror("pipe");
20          exit(1);
21      }
22
23      switch (pid = fork()) {
24          case -1 :
25              perror("fork");
26              exit(1);
27              break;
28          case 0 : /* child */
29              close(fd1[1]);
30              close(fd2[0]);
31              write(1, "Child Process:", 15);
32              len = read(fd1[0], buf, 256);
33              write(1, buf, len);
34
35              strcpy(buf, "Good\n");
36              write(fd2[1], buf, strlen(buf));
37              break;
38          default :
39              close(fd1[0]);
40              close(fd2[1]);
41              buf[0] = '\0';
42              write(fd1[1], "Hello\n", 6);
43
44              write(1, "Parent Process:", 15);
45              len = read(fd2[0], buf, 256);
46              write(1, buf, len);
47              waitpid(pid, &status, 0);
48              break;
49      }
50
51      return 0;
52  }
```

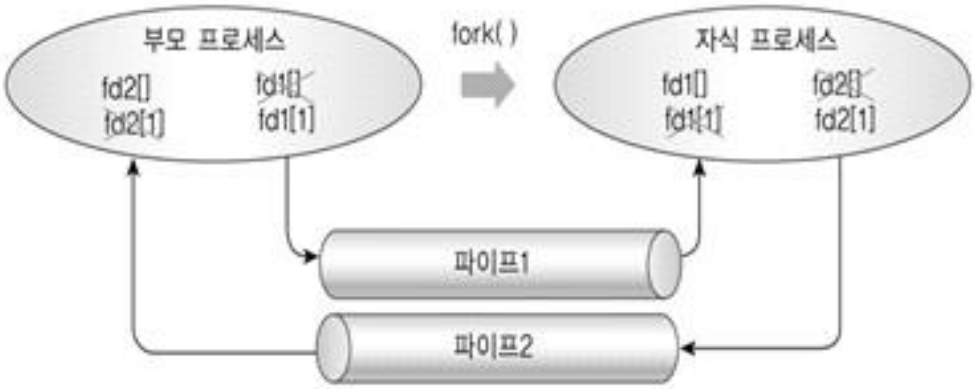
파이프 2개를 생성하기 위해 배열2개 선언

파이프 2개 생성

자식 프로세스
-fd1[0]으로 읽기
-fd2[1]로 쓰기

부모 프로세스
-fd1[1]로 쓰기
-fd2[0]으로 읽기

```
# ex9_5.out
Child Process:Hello
Parent Process:Good
```



【학습정리】

1. 이름 없는 파이프

* 파이프의 개념

- 두 프로세스간에 통신할 수 있도록 해주는 특수 파일
- 그냥 파이프라고 하면 일반적으로 이름 없는 파이프를 의미
- 이름 없는 파이프는 부모-자식 프로세스 간에 통신할 수 있도록 해줌
- 파이프는 기본적으로 단방향

2. 이름 없는 파이프 관련 함수

기능	함수원형
간단한 파이프 생성	FILE *popen(const char *command, const char *mode); int pclose(FILE *stream);
복잡한 파이프 생성	int pipe(int fildes[2]);