실습 11주차

CPS LAB

파이프

목차

프로세스간 통신 (파이프)

- 파이프(pipe)는 Unix 계열 운영체제에서 프로세스 간 통신을 위해 강력한 도구이다.

- 파이프는 두 프로세스 사이에서 한 방향으로 통신할 수 있도록 지원
  터미널 커맨드에서도 쉽게 사용 가능

- 생산자 – 소비자 패턴
  - 익명 파이프 (anonymous pipe)
  - 네임드 파이프 (named pipe)

익명 파이프

- 익명 파이프(anonymous pipe)는 부모 프로세스와 자식 프로세스 간 통신을 위해 사용
- popen의 경우 호출시 fork로 자식 프로세스를 생성하고 커맨드에서 지정한 명령을 exec 함수로 실행해 자식 프로세스가 수행하게 함.
- pipe의 경우 fildes 2개를 이용해 직접 pipe에 접근하여 작업가능

네임 파이프

- 네임 파이프(Named pipe)는 부모-자식 프로세스 간 통신하는 것이 아님,
  독립적인 프로세스 간 파일처럼, 데이터 주고 받는 형태로 하고 싶다면, 이름을 있어야 함

- FIFO(First In, First Out) 라는 특수한 파일 유형을 도입했다
  - 프로세스는 mknod() 시스템 콜을 사용하여, FIFO 생성함

#include <stdio.h>

FILE *popen(const char *command, const char *type);

Return Value:
on success, returns a pointer to an open stream that can be used to read or write to the pipe; if the fork(2) or pipe(2) calls fail, or if the function cannot allocate memory, NULL is returned.

예제풀이

읽기 전용

popen함수를 이용하여
“ls”명령어를 실행해,
각 출력내용을 읽어 받아,
printf로 다시 출력

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

int main(void) {

        FILE *fp;
        char buffer[1024];

        // 프로세스 "ls"라는 명령어 실행
        fp = popen("ls", "r");
        if( fp == NULL ) {
                perror("popen");
                exit(1);
        }

        // 명령어 실행하는 대로,
        // 여기서 fp를 읽어 받는다
        while( fgets(buffer, sizeof(buffer), fp) != NULL )
                printf("%s", buffer);

        pclose(fp);

        return 0;
}
```

```
(base) kyhooon@kyh:~/sysprogram_practice/p_ch10$ ./ex_1.out
ch10_1.c
ch10_1.out
ch10_2.c
ch10_2.out
ch10_3.c
ch10_3.out
ch10_4.c
ch10_4.out
ch10_5.c
ch10_5.out
ch10_6.c
ch10_6.out
ex_1.c
ex_1.out
kyh-FIFO
sys-FIFO
(base) kyhooon@kyh:~/sysprogram_practice/p_ch10$
```

예제풀이

쓰기 전용

popen함수를 이용하여
“wc”명령어에 대해 파이프를 열고,
쓰기모드로 설정함

줄, 단어, 문자 수 출력함

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

int main() {
        FILE *fp;
        char buffer[1024];

        fp = popen("wc", "w");
        if (fp == NULL) {
                perror("popen failed");
                exit(1);
        }

        fprintf(fp, "Hello\nWorld\n");

        pclose(fp);

        return 0;
}
```

```
(base) kyhooon@kyh:~/sysprogram_practice/p_ch10$ ./ex_2.out
      2       2      12
(base) kyhooon@kyh:~/sysprogram_practice/p_ch10$
```

```
#include <unistd.h>

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

Description:
Used for interprocess communication
The array pipefd is used to return two file descriptors referring to the ends of the pipe
pipefd[0] refers to the read end of the pipe
pipefd[1] refers to the write end of the pipe

Return Value:
On success, 0 is returned.
On error, -1 is returned

그림 10-1 pipe() 함수를 이용한 파이프 생성

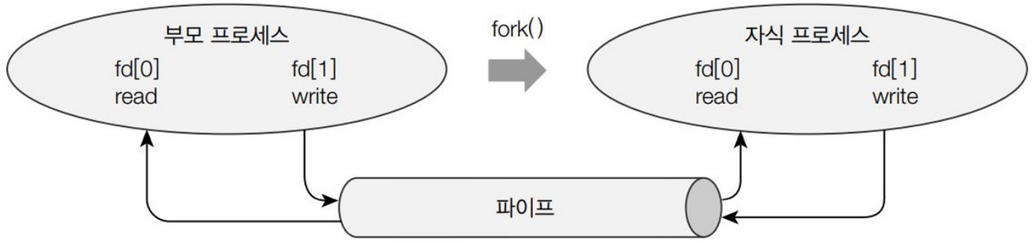


그림 10-2 자식 프로세스로 파일 기술자 복사

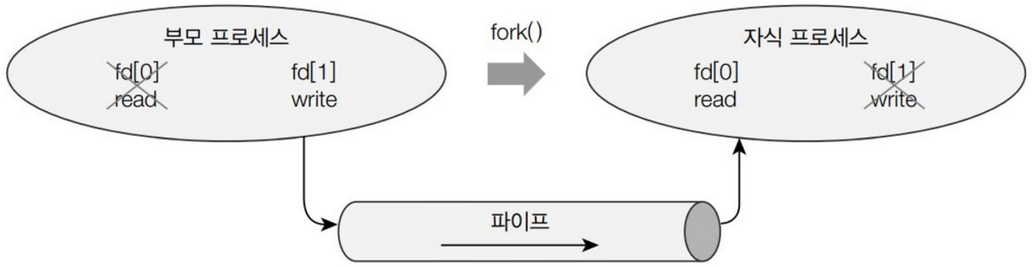


그림 10-3 부모 → 자식 방향으로 통신

예제풀이

예제 10-3

pipe()함수로 통신하기

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main(void) {

        int fd[2];
        pid_t pid;
        char buf[257];
        int len, status;

        if( pipe(fd) == -1 ) {
                perror("pipe");
                exit(1);
        }

        switch( pid = fork() ) {

                case -1 :
                        perror("fork");
                        exit(1);
                        break;

                case 0 :
                        // 자식프로세스
                        close(fd[1]);
                        write(1, "child process:", 15);
                        len = read(fd[0], buf, 256);
                        write(1, buf, len);
                        close(fd[0]);
                        break;

                default :
                        close(fd[0]);
                        write(fd[1], "Test Message\n", 14);
                        close(fd[1]);
                        waitpid(pid, &status, 0);
                        break;
        }

        return 0;
}
```

```
(base) kyhooon@kyh:~/sysprogram_practice/p_ch10$ ./ch10_3.out
child process:Test Message
(base) kyhooon@kyh:~/sysprogram_practice/p_ch10$
```

예제풀이

예제 10-4

pipe()함수로 명령 실행하기

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main(void) {

        int fd[2];
        pid_t pid;

        if( pipe(fd) == -1 ) {
                perror("pipe");
                exit(1);
        }

        switch( pid = fork() ) {

                case -1 :
                        perror("fork");
                        exit(1);
                        break;

                case 0 :
                        // 자식 프로세스
                        close(fd[1]);
                        if( fd[0] != 0 ) {
                                dup2(fd[0], 0);
                                close(fd[0]);
                        }
                        execlp("grep", "grep", "ssh", (char*)NULL);
                        exit(1);
                        break;

                default :

                        close(fd[0]);
                        if( fd[1] != 1 ) {
                                dup2(fd[1], 1);
                                close(fd[1]);
                        }
                        execlp("ps", "ps", "-ef", (char *)NULL);
                        wait(NULL);
                        break;
        }

        return 0;
}
```

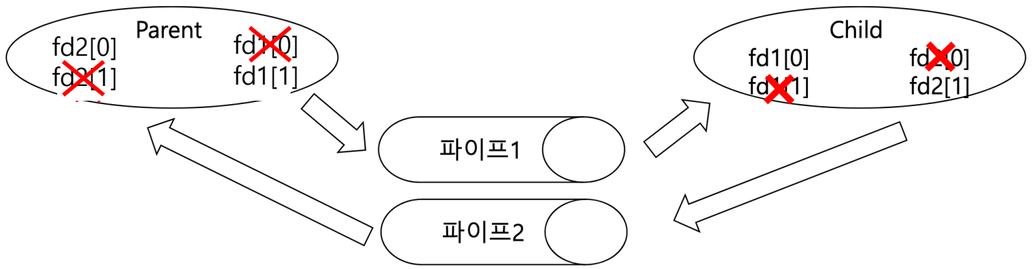
```
(base) kyhooon@kyh:~/sysprogram_practice/p_ch10$ ./ch10_4.out
root        1174       1  0 10월 18 ?      00:00:00 sshd: /usr/sbin/sshd
kyhooon     2039    1972  0 10월 18 ?      00:00:15 /usr/bin/ssh-agent /u
kyhooon    67142    1874  0 10월 26 ?      00:00:00 /usr/bin/ssh-agent -D
root      829470    1174  0 13:44 ?        00:00:00 sshd: kyhooon [priv]
kyhooon   829569  829470  0 13:44 ?        00:00:00 sshd: kyhooon@pts/1
kyhooon   830329  830328  0 14:45 pts/4    00:00:00 grep ssh
(base) kyhooon@kyh:~/sysprogram_practice/p_ch10$
```

예제풀이

예제 10-5

pipe()함수로 양방향 통신하기

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <string.h>

int main(void) {

        int fd1[2], fd2[2];
        pid_t pid;
        char buf[257];
        int len, status;

        if( pipe(fd1) == -1 ) {
                perror("pipe");
                exit(1);
        }

        if( pipe(fd2) == -1 ) {
                perror("pipe");
                exit(1);
        }

        switch( pid = fork() ) {

                case -1 :
                        perror("fork");
                        exit(1);
                        break;

                case 0 :
                        // 자식 프로세스
                        close(fd1[1]);
                        close(fd2[0]);
                        len = read(fd1[0], buf, 256);
                        write(1, "child process:", 15);
                        write(1, buf, len);

                        strcpy(buf, "Good\n");
                        write(fd2[1], buf, strlen(buf));
                        break;

                default :
                        close(fd1[0]);
                        close(fd2[1]);
                        write(fd1[1], "Hello\n", 6);

                        len = read(fd2[0], buf, 256);
                        write(1, "parent process:", 15);
                        write(1, buf, len);
                        waitpid(pid, &status, 0);
                        break;
        }

        return 0;
}
```



연습 문제

실습#

채팅 프로그램
파이프를 이용해 메시지를 한 번씩 주고 받는 프로그램 작성,
파이프 함수를 이용해 2개의 파이프 생성 후,
Child -> Parent / Parent -> Child으로 사용

감사합니다.

CPS LAB