응용과제 13
pipe를 이용한 주문 - 조리 프로그램을 개선하여 부모 측에서 주문/완료 내용을 별도 파일에 저장하도록 개선

(응용 12번과 while문에서 log_now함수를 호출하는 부분만 다르기에 while문 부분만 캡쳐하였습니다.)

```
while(1)
{
        struct sigaction act; //메뉴 등장 전 인터럽트 발생시 핸들러처리하도록
        sigemptyset(&act.sa_mask);
        act.sa_flags = 0;
        act.sa_handler = int_handler;
        if(sigaction(SIGINT, &act, (struct sigaction *)NULL) < 0)
        {
                perror("sigaction 에러 발생");
                exit(3);
        }

        printf("[부모] 주문하실 메뉴를 입력해주세요 (문자열) : ");
        fgets(menu, MENU_SIZE, stdin);
        if(write(c2k_fd[1], menu, strlen(menu)) != -1) //파이프에 쓰기성공시
        {
                printf("[부모] 메뉴를 전달하였습니다.\n\n");
                log_now(0, menu); //로그기록
        }
        else
        {
                perror("write 에러 발생");
                exit(4);
        }

        act.sa_handler = SIG_IGN; //요리 완료전 SIGINT, SIGQUIT 무시
        if(sigaction(SIGINT, &act, (struct sigaction *)NULL) < 0)
        {
                perror("sigaction 에러 발생");
                exit(3);
        }
        if(sigaction(SIGQUIT, &act, (struct sigaction *)NULL) < 0)
        {
                perror("sigaction 에러 발생");
                exit(3);
        }

        //after child process

        char buf[BUFSIZ];
        if(read(k2c_fd[0], buf, BUFSIZ) != -1) //자식에게 파이프로 완료를 전달받았을때
        {
                if(strncmp("success", buf, 7) == 0)
                {
                        printf("[부모] 메뉴를 전달받아 손님에게 제공하였습니다.\n");
                        log_now(1, menu); //로그기록
                }
        }
        else
        {
                perror("read 에러 발생");
                exit(5);
        }
}
```

log_now 함수

```
void log_now(int select, char menu[])
{

        int time_fd;
        time_t t;
        struct tm tm;
        char logbuf[BUFSIZ];

        //로그파일을 남기기위한 log.txt를 open
        time_fd = open("log.txt", O_WRONLY | O_CREAT | O_APPEND , 0644 );
        if(time_fd == -1)
        {
                perror("open log file");
                exit(6);
        }

        t = time(NULL);
        tm = *localtime(&t);
        if(select == 0) //주문전달
        {
                sprintf(logbuf, "%d년 %d월 %d일 %d시 %d분 %d초 주문전달 메뉴 : %s",
                                tm.tm_year+1900, tm.tm_mon+1, tm.tm_mday, tm.tm_hour,
                                tm.tm_min, tm.tm_sec, menu);
        }
        else if(select == 1)//주문완료
        {
                sprintf(logbuf, "%d년 %d월 %d일 %d시 %d분 %d초 주문완료\n",
                                tm.tm_year+1900, tm.tm_mon+1, tm.tm_mday, tm.tm_hour,
                                tm.tm_min, tm.tm_sec);
        }
        write(time_fd, logbuf, strlen(logbuf));
        close(time_fd);
}
```

로그파일을 열고 할당할 time_fd를 선언합니다. 그후 시간을 기록하기위해 time.h에 정의되어있는 time타입 t와 tm 구조체 tm을 선언합니다. 그후 sprintf를 사용하여 open한 log.txt에 기록해야하므로 문자열을 담을 logbuf를 선언해줍니다. open함수를 사용하여 log.txt를 열어줍니다. 기록용으로 사용할것이므로 쓰기전용으로 열어주고, 없을시 생성옵션인 CREAT과 기존파일을 지우지않고 계속하여 추가될수있도록 APPEND도 사용해주고 권한도 0644로 열어줍니다. 그후 time함수를 사용하여 현재 시간을 t에 저장해줍니다. 그후 localtime 함수를 사용하여 저장하였던 현재시간을 tm 구조체에 지역시간 기준으로 값을 초기화해줍니다. 함수의 매개변수로 들어간 0과 1의경우는 주문 전달과 주문완료를 다르게 출력하기위해 작성한것이고, sprintf를 사용하여 logbuf에 tm구조체의 멤버를 접근하여 기록해줍니다. 그 후 write함수를 사용하여 time_fd(log.txt의 파일 디스크립터) logbuf를 쓰기 해줍니다. 모든 작업이 끝나면 open하였던 time_fd를 닫아줌으로써 펑션이 종료됩니다.

소스코드 (counter.c 전체)

```
#include<unistd.h>
#include<signal.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<fcntl.h>
#include<time.h>

#define MENU_SIZE 128

int pid_child = -1;

void log_now(int select, char menu[]);

void int_handler(int signo)
{
	kill(pid_child, SIGINT); //자식종료
	printf("\n\n[부모] 자식 프로세스를 종료하였습니다.\n");
	printf("[부모] 자신을 종료합니다.\n");
	exit(0);
}

int main(void)
{
	int c2k_fd[2];
	int k2c_fd[2];

	if(pipe(c2k_fd) == -1)
	{
		perror("pipe");
		exit(1);
	}
	if(pipe(k2c_fd) == -1)
	{
		perror("pipe");
		exit(1);
	}

	pid_child = fork();

	if(pid_child < 0) //fork 오류
	{
		perror("fork");
		exit(2);
	}
	else if(pid_child == 0) //자식프로세스
	{
		close(c2k_fd[1]);
		close(k2c_fd[0]);

		dup2(c2k_fd[0], 0); // stdin -> c2k_fd[0]
		dup2(k2c_fd[1], 1); // stdout -> k2c_fd[1]
```

```
        if(execl("./kitchen.exe", "kitchen.exe", NULL) < 0)
        {
                perror("execl");
                exit(2);
        }
        close(c2k_fd[0]);
        close(k2c_fd[1]);
}
else
{
        close(c2k_fd[0]);
        close(k2c_fd[1]);

        char menu[MENU_SIZE];

        while(1)
        {
                struct sigaction act; //메뉴 동작 전 인터럽트 발생시 핸들러처리하도록
                sigemptyset(&act.sa_mask);
                act.sa_flags = 0;
                act.sa_handler = int_handler;
                if(sigaction(SIGINT, &act, (struct sigaction *)NULL) < 0)
                {
                        perror("sigaction 에러 발생");
                        exit(3);
                }

                printf("[부모] 주문하실 메뉴를 입력해주세요 (문자열)  : ");
                fgets(menu, MENU_SIZE, stdin);
                if(write(c2k_fd[1], menu, strlen(menu)) != -1) //파이프에 쓰기성공시
                {
                        printf("[부모] 메뉴를 전달하였습니다.\n\n");
                        log_now(0, menu); //로그기록
                }
                else
                {
                        perror("write 에러 발생");
                        exit(4);
                }

                act.sa_handler = SIG_IGN; //요리 완료전 SIGINT, SIGQUIT 무시
                if(sigaction(SIGINT, &act, (struct sigaction *)NULL) < 0)
                {
                        perror("sigaction 에러 발생");
                        exit(3);
                }
                if(sigaction(SIGQUIT, &act, (struct sigaction *)NULL) < 0)
                {
                        perror("sigaction 에러 발생");
                        exit(3);
                }

                //after child process
```

```
                        char buf[BUFSIZ];
                        if(read(k2c_fd[0], buf, BUFSIZ) != -1) //자식에게 파이프로 완료를 전달받았을때
                        {
                                if(strncmp("success", buf, 7) == 0)
                                {
                                        printf("[부모] 메뉴를 전달받아 손님에게 제공하였습니다.\n");
                                        log_now(1, menu); //로그기록
                                }
                        }
                        else
                        {
                                perror("read 에러 발생");
                                exit(5);
                        }
                }
                close(c2k_fd[1]);
                close(k2c_fd[0]);
        }
        return 0;
}

void log_now(int select, char menu[])
{
        int time_fd;
        time_t t;
        struct tm tm;
        char logbuf[BUFSIZ];

        //로그파일을 남기기위한 log.txt를 open
        time_fd = open("log.txt", O_WRONLY | O_CREAT | O_APPEND , 0644 );
        if(time_fd == -1)
        {
                perror("open log file");
                exit(6);
        }

        t = time(NULL);
        tm = *localtime(&t);
        if(select == 0) //주문전달
        {
                sprintf(logbuf, "%d년 %d월 %d일 %d시 %d분 %d초 주문전달 메뉴 : %s",
                                tm.tm_year+1900, tm.tm_mon+1, tm.tm_mday, tm.tm_hour,
                                tm.tm_min, tm.tm_sec, menu);
        }
        else if(select == 1)//주문완료
        {
                sprintf(logbuf, "%d년 %d월 %d일 %d시 %d분 %d초 주문완료\n",
                                tm.tm_year+1900, tm.tm_mon+1, tm.tm_mday, tm.tm_hour,
                                tm.tm_min, tm.tm_sec);
        }
        write(time_fd, logbuf, strlen(logbuf));
        close(time_fd);
}
```

출력결과

```
[khm970514@lily ch7]$ cat log.txt
cat: log.txt: 그런 파일이나 디렉터리가 없습니다
[khm970514@lily ch7]$ ./counter.exe
[부모] 주문하실 메뉴를 입력해주세요 (문자열) : 김밥
[부모] 메뉴를 전달하였습니다.

[자식] 주문을 전달받았습니다. 메뉴 : 김밥
[자식] 조리를 시작합니다.
[자식] 완성된 요리를 전달합니다.

[부모] 메뉴를 전달받아 손님에게 제공하였습니다.
[부모] 주문하실 메뉴를 입력해주세요 (문자열) : 치킨
[부모] 메뉴를 전달하였습니다.
[자식] 주문을 전달받았습니다. 메뉴 : 치킨

[자식] 조리를 시작합니다.
[자식] 완성된 요리를 전달합니다.

[부모] 메뉴를 전달받아 손님에게 제공하였습니다.
[부모] 주문하실 메뉴를 입력해주세요 (문자열) : 햄버거
[부모] 메뉴를 전달하였습니다.

[자식] 주문을 전달받았습니다. 메뉴 : 햄버거
[자식] 조리를 시작합니다.
[자식] 완성된 요리를 전달합니다.

[부모] 메뉴를 전달받아 손님에게 제공하였습니다.
[부모] 주문하실 메뉴를 입력해주세요 (문자열) : ^C

[부모] 자식 프로세스를 종료하였습니다.
[부모] 자신을 종료합니다.
[khm970514@lily ch7]$ cat log.txt
2019년 12월 4일 20시 34분 22초 주문전달 메뉴 : 김밥
2019년 12월 4일 20시 34분 27초 주문완료
2019년 12월 4일 20시 34분 28초 주문전달 메뉴 : 치킨
2019년 12월 4일 20시 34분 33초 주문완료
2019년 12월 4일 20시 34분 53초 주문전달 메뉴 : 햄버거
2019년 12월 4일 20시 34분 58초 주문완료
[khm970514@lily ch7]$
```