응용과제 11

Makefile

```
# Makefile

cc = gcc
CFLAG = -g

all :
        make counter.exe
        make kitchen.exe

counter.exe : counter.c
        gcc -o counter.exe counter.c

kitchen.exe : kitchen.c
        gcc -o kitchen.exe kitchen.c

~
```

```
[khm970514@lily ch7]$ make
make counter.exe
make[1]: Entering directory `/home/2019/UNIX/khm970514/unix/ch7'
make[1]: `counter.exe'는 이미 갱신되었습니다.
make[1]: Leaving directory `/home/2019/UNIX/khm970514/unix/ch7'
make kitchen.exe
make[1]: Entering directory `/home/2019/UNIX/khm970514/unix/ch7'
make[1]: `kitchen.exe'는 이미 갱신되었습니다.
make[1]: Leaving directory `/home/2019/UNIX/khm970514/unix/ch7'
[khm970514@lily ch7]$
```

counter.c

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

int pid_child;
int count = 0;

void int_handler(int signo)
{
        kill(pid_child, SIGINT);
        wait();
        printf("\n[부모] 자식프로세스를 종료하였습니다.\n");
        exit(0);
}

void sigusr1_handler(int signo)
{
        printf("\n[부모] 요리를 전달받아 손님에게 제공하였습니다.\n");
        count--;
        printf("[부모] 현재 진행중인 주문 : %d\n", count);
        if(count == 0)
        {
                struct sigaction act;
                sigemptyset(&act.sa_mask);
                act.sa_flags = 0;
                act.sa_handler = int_handler;
                if(sigaction(SIGINT, &act, (struct sigaction *)NULL) < 0)
                {
                        perror("sigaction error");
                        exit(3);
                }

                act.sa_handler = SIG_DFL;
                if(sigaction(SIGUSR1, &act, (struct sigaction *)NULL) < 0)
                {
                        perror("sigaction error");
                        exit(3);
                }
        }
}

int main(void)
{
        pid_child = fork();
        if(pid_child < 0)
        {
                perror("fork");
                exit(1);
        }
        else if(pid_child == 0)
        {
                if(execl("./kitchen.exe", "kitchen.exe", NULL) < 0)
                {
                        perror ("execl");
                        exit(2);
                }
        }
```

```
	else
	{
		char answer;
		struct sigaction act;
		act.sa_flags = 0;
		act.sa_handler = int_handler;
		sigaction(SIGINT, &act, (struct sigaction *)NULL);

		while(1)
		{
			printf("[부모] 음식을 주문하시겠습니까?"
				"\n1.피자 2.치킨 (숫자입력, 종료 0) : ");
			fgets(&answer, sizeof(answer), stdin);
			__fpurge(stdin);
			//scanf("%c", &answer);
			//getchar();
			answer = getc(stdin);
			switch(answer)
			{
				case '0' :
					printf("[부모] 프로그램을 종료합니다.\n");
					exit(0);
					break;
				case '1' :
					if(count == 0)
					{
						act.sa_handler = SIG_IGN;
						if(sigaction(SIGINT, &act, (struct sigaction *)NULL) < 0)
						{
							perror("sigaction error");
							exit(3);
						}

						act.sa_handler = sigusr1_handler;
						if(sigaction(SIGUSR1, &act, (struct sigaction *)NULL) < 0)
						{
							perror("sigaction error");
							exit(3);
						}
					}
					count++;
					kill (pid_child, SIGUSR1);
					printf("\n[부모] 피자주문을 전달합니다.\n");
					break;
				case '2' :
					if(count == 0)
					{
						act.sa_handler = SIG_IGN;
						if(sigaction(SIGINT, &act, (struct sigaction *)NULL) < 0)
						{
							perror("sigaction error");
							exit(3);
						}

						act.sa_handler = sigusr1_handler;
						if(sigaction(SIGUSR1, &act, (struct sigaction *)NULL) < 0)
						{
							perror("sigaction error");
							exit(3);
						}
					}
					count++;
					kill (pid_child, SIGUSR2);
					printf("\n[부모] 치킨주문을 전달합니다.\n");
					break;
			}
		}
	}
	return 0;
}
```

소스코드

```
#include <sys/types.h>
#include <unistd.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>

int pid_child;
int count = 0;

void int_handler(int signo)
{
        kill(pid_child, SIGINT);
        wait();
        printf("₩n[부모] 자식프로세스를 종료하였습니다.₩n");
        exit(0);
}

void sigusr1_handler(int signo)
{
        printf("₩n[부모] 요리를 전달받아 손님에게 제공하였습니다.₩n");
        count--;
        printf("[부모] 현재 진행중인 주문 :  %d₩n", count);
        if(count == 0)
        {
                struct sigaction act;
                sigemptyset(&act.sa_mask);
                act.sa_flags = 0;
                act.sa_handler = int_handler;
                if(sigaction(SIGINT, &act, (struct sigaction *)NULL) < 0)
                {
                        perror("sigaction error");
                        exit(3);
                }

                act.sa_handler = SIG_DFL;
                if(sigaction(SIGUSR1, &act, (struct sigaction *)NULL) < 0)
                {
                        perror("sigaction error");
                        exit(3);
                }
        }
}

int main(void)
{
        pid_child = fork();
        if(pid_child < 0)
        {
```

```
        perror("fork");
        exit(1);
}
else if(pid_child == 0)
{
        if(execl("./kitchen.exe", "kitchen.exe", NULL) < 0)
        {
                perror ("execl");
                exit(2);
        }
}
else
{
        char answer;
        struct sigaction act;
        act.sa_flags = 0;
        act.sa_handler = int_handler;
        sigaction(SIGINT, &act, (struct sigaction *)NULL);

        while(1)
        {
                printf("[부모] 음식을 주문하시겠습니까?"
                        "₩n1.피자 2.치킨  (숫자입력, 종료 0) : ");
                fgets(&answer, sizeof(answer), stdin);
                __fpurge(stdin);
                //scanf("%c", &answer);
                //getchar();
                answer = getc(stdin);
                switch(answer)
                {
                        case '0' :
                                printf("[부모] 프로그램을 종료합니다.₩n");
                                exit(0);
                                break;
                        case '1' :
                                if(count == 0)
                                {
                                        act.sa_handler = SIG_IGN;
                                        if(sigaction(SIGINT, &act, (struct sigaction *)NULL) < 0)
                                        {
                                                perror("sigaction error");
                                                exit(3);
                                        }

                                        act.sa_handler = sigusr1_handler;
                                        if(sigaction(SIGUSR1, &act, (struct sigaction *)NULL) < 0)
                                        {
                                                perror("sigaction error");
                                                exit(3);
                                        }
```

```
                        }
                        count++;
                        kill (pid_child, SIGUSR1);
                        printf("₩n[부모] 피자주문을  전달합니다.₩n");
                        break;
                case '2' :
                        if(count == 0)
                        {
                                act.sa_handler = SIG_IGN;
                                if(sigaction(SIGINT, &act, (struct sigaction *)NULL) < 0)
                                {
                                        perror("sigaction error");
                                        exit(3);
                                }

                                act.sa_handler = sigusr1_handler;
                                if(sigaction(SIGUSR1, &act, (struct sigaction *)NULL) < 0)
                                {
                                        perror("sigaction error");
                                        exit(3);
                                }
                        }
                        count++;
                        kill (pid_child, SIGUSR2);
                        printf("₩n[부모] 치킨주문을  전달합니다.₩n");
                        break;
                }
        }
    }
    return 0;
}
```

kitchen.c

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

#define TIME_FOR_PIZZA 10
#define TIME_FOR_CHICKEN 15

typedef struct in_progress {
        int time;
        struct in_progress *next;
}in_progress_t;

in_progress_t *header = NULL;
void sigusr1_handler(int signo);

void alrm_handler(int signo)
{
        in_progress_t *tmp;
        tmp = header->next;
        int all_time = 0;
        while(tmp != NULL)
        {
                all_time = all_time + tmp->time;
                printf("\n[자식] 현재 진행중 요리 남은시간 : %d\n", all_time);
                tmp = tmp->next;
        }
        free(tmp);

        tmp = header;
        if(header->next)
        {
                header = header->next;
                alarm(header->time);
        }

        else
        {
                header = NULL;
                struct sigaction act;
                sigemptyset(&act.sa_mask);
                act.sa_flags = 0;
                act.sa_handler = SIG_DFL;
                if(sigaction(SIGINT, &act, (struct sigaction *)NULL) < 0)
                {
                        perror("sigaction error");
                        exit(3);
                }

                act.sa_handler = SIG_IGN;
                if(sigaction(SIGALRM, &act, (struct sigaction *)NULL) < 0)
                {
                        perror("sigaction eroor");
                        exit(3);
                }
        }

        kill(getppid(), SIGUSR1);
        free(tmp);
}
```

```c
void sigusr1_handler(int signo)
{
        in_progress_t *tmp;
        in_progress_t *prev;
        in_progress_t *new;

        printf("\n[자식] 피자주문 도착\n");

        new = malloc(sizeof(in_progress_t));
        new->time = TIME_FOR_PIZZA;
        new->next = NULL;

        if(header == NULL)
        {
                struct sigaction act;
                sigemptyset(&act.sa_mask);
                act.sa_flags = 0;
                act.sa_handler = SIG_IGN;
                if(sigaction(SIGINT, &act, (struct sigaction *)NULL) < 0)
                {
                        perror("sigaction error");
                        exit(3);
                }

                act.sa_handler = alrm_handler;
                if(sigaction(SIGALRM, &act, (struct sigaction *)NULL) < 0)
                {
                        perror("sigaction error");
                        exit(3);
                }
                header = new;
        }
        else
        {
                header->time = alarm(0);

                for(tmp = header; tmp; prev = tmp, tmp = tmp->next)
                {
                        new->time = new->time - tmp->time;
                }
                if(new->time <= 0)
                {
                        new->time = 1;
                }
                prev->next = new;
        }

        printf("\n");
        in_progress_t *left_time;
        int all_time = 0;
        left_time = header;
        while(left_time != NULL)
        {
                all_time = all_time + left_time->time;
                printf("[자식] 남은요리시간 : %d\n", all_time);
                left_time = left_time->next;
        }
        alarm(header->time);
}
```

```c
void sigusr2_handler(int signo)
{
        in_progress_t *tmp;
        in_progress_t *prev;
        in_progress_t *new;

        printf("\n[자식] 치킨주문 도착\n");

        new = malloc(sizeof(in_progress_t));
        new->time = TIME_FOR_CHICKEN;
        new->next = NULL;

        if(header == NULL)
        {
                struct sigaction act;
                sigemptyset(&act.sa_mask);
                act.sa_flags = 0;
                act.sa_handler = SIG_IGN;
                if(sigaction(SIGINT, &act, (struct sigaction *)NULL) < 0)
                {
                        perror("sigaction error");
                        exit(3);
                }

                act.sa_handler = alrm_handler;
                if(sigaction(SIGALRM, &act, (struct sigaction *)NULL) < 0)
                {
                        perror("sigaction error");
                        exit(3);
                }
                header = new;
        }
        else
        {
                header->time = alarm(0);

                for(tmp = header; tmp; prev = tmp, tmp = tmp->next)
                {
                        new->time = new->time - tmp->time;
                }
                if(new->time <= 0)
                {
                        new->time = 1;
                }
                prev->next = new;
        }

        printf("\n");
        in_progress_t *left_time;
        int all_time = 0;
        left_time = header;
        while(left_time != NULL)
        {
                all_time = all_time + left_time->time;
                printf("[자식] 남은요리시간 : %d\n", all_time);
                left_time = left_time->next;
        }

        alarm(header->time);
}
```

```c
int main(void)
{
        struct sigaction act;
        sigemptyset(&act.sa_mask);
        act.sa_flags = 0;

        act.sa_handler = sigusr1_handler;
        if(sigaction(SIGUSR1, &act, (struct sigaction *)NULL) < 0)
        {
                perror("sigaction error");
                exit(3);
        }

        act.sa_handler = sigusr2_handler;
        if(sigaction(SIGUSR2, &act, (struct sigaction *)NULL) < 0)
        {
                perror("sigaction error");
                exit(3);
        }

        while(1)
        {
                sleep(999);
        }
}
```

소스코드

```
#include <sys/types.h>
#include <unistd.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>

#define TIME_FOR_PIZZA 10
#define TIME_FOR_CHICKEN 15

typedef struct in_progress {
        int time;
        struct in_progress *next;
}in_progress_t;

in_progress_t *header = NULL;
void sigusr1_handler(int signo);

void alrm_handler(int signo)
{
        in_progress_t *tmp;
        tmp = header->next;
        int all_time = 0;
        while(tmp != NULL)
        {
                all_time = all_time + tmp->time;
                printf("₩n[자식] 현재 진행중 요리 남은시간 :  %d₩n", all_time);
                tmp = tmp->next;
        }
        free(tmp);

        tmp = header;
        if(header->next)
        {
                header = header->next;
                alarm(header->time);
        }

        else
        {
                header = NULL;
                struct sigaction act;
                sigemptyset(&act.sa_mask);
                act.sa_flags = 0;
                act.sa_handler = SIG_DFL;
                if(sigaction(SIGINT, &act, (struct sigaction *)NULL) < 0)
                {
                        perror("sigaction error");
                        exit(3);
                }

                act.sa_handler = SIG_IGN;
```

```
                if(sigaction(SIGALRM, &act, (struct sigaction *)NULL) < 0)
                {
                        perror("sigaction eroor");
                        exit(3);
                }
        }

        kill(getppid(), SIGUSR1);
        free(tmp);
}

void sigusr1_handler(int signo)
{
        in_progress_t *tmp;
        in_progress_t *prev;
        in_progress_t *new;

        printf("₩n[자식] 피자주문 도착₩n");

        new = malloc(sizeof(in_progress_t));
        new->time = TIME_FOR_PIZZA;
        new->next = NULL;

        if(header == NULL)
        {
                struct sigaction act;
                sigemptyset(&act.sa_mask);
                act.sa_flags = 0;
                act.sa_handler = SIG_IGN;
                if(sigaction(SIGINT, &act, (struct sigaction *)NULL) < 0)
                {
                        perror("sigaction error");
                        exit(3);
                }

                act.sa_handler = alrm_handler;
                if(sigaction(SIGALRM, &act, (struct sigaction *)NULL) < 0)
                {
                        perror("sigaction error");
                        exit(3);
                }
                header = new;
        }
        else
        {
                header->time = alarm(0);

                for(tmp = header; tmp; prev = tmp, tmp = tmp->next)
                {
                        new->time = new->time - tmp->time;
                }
                if(new->time <= 0)
                {
                        new->time = 1;
```

```
                }
                prev->next = new;
        }

        printf("₩n");
        in_progress_t *left_time;
        int all_time = 0;
        left_time = header;
        while(left_time != NULL)
        {
                all_time = all_time + left_time->time;
                printf("[자식] 남은요리시간 : %d₩n", all_time);
                left_time = left_time->next;
        }
        alarm(header->time);
}

void sigusr2_handler(int signo)
{
        in_progress_t *tmp;
        in_progress_t *prev;
        in_progress_t *new;

        printf("₩n[자식] 치킨주문 도착₩n");

        new = malloc(sizeof(in_progress_t));
        new->time = TIME_FOR_CHICKEN;
        new->next = NULL;

        if(header == NULL)
        {
                struct sigaction act;
                sigemptyset(&act.sa_mask);
                act.sa_flags = 0;
                act.sa_handler = SIG_IGN;
                if(sigaction(SIGINT, &act, (struct sigaction *)NULL) < 0)
                {
                        perror("sigaction error");
                        exit(3);
                }

                act.sa_handler = alrm_handler;
                if(sigaction(SIGALRM, &act, (struct sigaction *)NULL) < 0)
                {
                        perror("sigaction error");
                        exit(3);
                }
                header = new;
        }
        else
        {
                header->time = alarm(0);

                for(tmp = header; tmp; prev = tmp, tmp = tmp->next)
```

```
                {
                        new->time = new->time - tmp->time;
                }
                if(new->time <= 0)
                {
                        new->time = 1;
                }
                prev->next = new;
        }

        printf("₩n");
        in_progress_t *left_time;
        int all_time = 0;
        left_time = header;
        while(left_time != NULL)
        {
                all_time = all_time + left_time->time;
                printf("[자식] 남은요리시간 : %d₩n", all_time);
                left_time = left_time->next;
        }

        alarm(header->time);
}

int main(void)
{
        struct sigaction act;
        sigemptyset(&act.sa_mask);
        act.sa_flags = 0;

        act.sa_handler = sigusr1_handler;
        if(sigaction(SIGUSR1, &act, (struct sigaction *)NULL) < 0)
        {
                perror("sigaction error");
                exit(3);
        }

        act.sa_handler = sigusr2_handler;
        if(sigaction(SIGUSR2, &act, (struct sigaction *)NULL) < 0)
        {
                perror("sigaction error");
                exit(3);
        }

        while(1)
        {
                sleep(999);
        }
}
```

이 프로그램은 응용과제 10번의 주문을 보내고 받아오는 사이에 중복으로 여러번 주문을 보낼수 없는 것을 해결한 프로그램입니다. 또한 메뉴를 선택하여 서로다른 메뉴를 주문할 수 있습니다.

부모에서 pause를 사용하여 자식에게 sigusr1을 받아오면 동작하던 부분을 삭제하여 부모가 자식에게 요리완성 시그널을 받지 못하여도 연속하여 주문할 수 있도록 하였습니다.

자식프로세스에서 메뉴마다 조리시간이 다르기 때문에 상수로 각각 2가지의 조리시간을 정해주었습니다.
그리고 부모가 SIGUSR1, SIGUSR2을 kill로 보내는 경우 (다른메뉴를 보내는 경우) 해당 시그널의 sa.handler를 정해주어 다른메뉴로 동작하도록 처리하였습니다.

연결리스트로 메뉴의 조리시간을 관리해야하기 때문에 구조체로 노드를 선언하고 멤버로 time과 다음 리스트를 가리킬 포인터 next를 선언해주었습니다. 부모가 시그널을 보내 각각의 핸들러로 동작하게되면 임시저장할 tmp와 이전을 나타낼 prev, 새로운 리스트인 new를 선언해주었습니다. malloc을 사용하여 새로운 리스트를 만들어주고 시간에 조리시간을 할당해줍니다. new에 연결될 다음 포인터는 널입니다.

만약 헤더가 널이면 인터럽트와 시그알람의 처리핸들러를 정해줍니다. 그후 새롭게 만든 new를 헤더로 지정해줍니다. 헤더가 널이아니면 time 에 alarm0을 넣어주는데 알람시그널을 받기전에 alarm을 호출하기 때문에 남은 초가 저장되게 됩니다. 그후 tmp를 사용하여 new->time의 값을 기존의 연결리스트가 있었으면 그전의 시간에서 현재 새로운 시간을 빼서 할당하도록 합니다. 이렇게하면 연결리스트에 남은 time 값들이 기존에 들어왔던 주문시간 - 현재 새로들어운 주문시간으로 재설정되어 alrm에 매개변수로 들어가므로 alrm시간을 설정할 수 있게됩니다.

현재 조리중인 음식을 출력하기위해서 left_time을 새로 선언하여 사용하였고 alarm핸들러에서는 이미 사용되고있던 tmp를 사용해주었습니다. 새롭게 선언한 구조체에 헤더를 할당하고 헤더가 널이아닌경우를 루프시킵니다. all_time 변수를 선언해주어서 누적되어야하는 현재남은 조리시간을 alltime=alltime+lefttime->time으로 할당해주어서 출력해주었습니다. 이렇게 하지 않으면 리스트에있는 time값이 출력되기 때문에 실제 남은 시간과는 다르게 출력되기 때문입니다. 나머지부분은 응용10번과 같습니다.

문제점 : 리스트의 말단에 새주문을 받아 시간을 넣기 때문에 이 프로그램에서 피자, 치킨, 피자 이런식으로 주문을 넣게되면 new->time이 음수값이 되어 주문을 빠르게 넣는경우처럼 시간이 1로 고정되어버립니다. (추후 응용과제에서 리스트의 중간에 넣는다던지 앞에 넣는다던지 하는 방법으로 해결 예정)

확인표 작성

확인사항	완성여부	비고
1. (부모) 자식 프로세스 생성 및 exec 정상 작동 여부	O	
2. (부모) 주문 받기 전 Ctrl-C 받았을 때 정상 동작(메시지 출력 / 자식프로세스 종료 / 자신 종료) 여부	O	
3. (부모) 조리 완료 전 3개 이상 주문 받아 자식에게 전달 / 순차적으로 완성된 주문에 대한 완료 메시지 출력	O	
4. 모든 주문 완료 후 Ctrl-C 받았을 때 자식 종료시키고 종료	O	
5. (부모) 조리 완료 전 3개 이상 주문 받아 자식에게 전달 / 완료 전 Ctrl-C 받았을 때 무시	O	
6. (자식) 주문 받았을 때 조리 중인 리스트에 추가하고 화면에 현재 조리 중인 각 요리 alarm 시간 출력	O	
7. (자식) SIGALARM 받았을 때 리스트에서 제거하고, 현재 조리 중인 각 요리 alarm 시간 출력 / 부모에게 통보	O	

1. 정상작동 여부

```
[khm970514@lily ch7]$ ./counter.exe
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) : 1

[부모] 피자주문을 전달합니다.
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[자식] 피자주문 도착

[자식] 남은요리시간 : 10

[부모] 요리를 전달받아 손님에게 제공하였습니다.
[부모] 현재 진행중인 주문 : 0
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) : 0
[부모] 프로그램을 종료합니다.
[khm970514@lily ch7]$
```

2. 주문받기전 ctrl-c 받는 경우

```
[khm970514@lily ch7]$ ./counter.exe
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) : ^C
[부모] 자식프로세스를 종료하였습니다.
[khm970514@lily ch7]$
```

3. 조리완료전 3개이상 주문받아 자식에게 전달 / 순차적으로 완료된 주문에 대한 메시지 출력

```
[khm970514@lily ch7]$ ./counter.exe
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) : 1

[부모] 피자주문을 전달합니다.
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[자식] 피자주문 도착

[자식] 남은요리시간 : 10
1

[부모] 피자주문을 전달합니다.
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[자식] 피자주문 도착

[자식] 남은요리시간 : 8
[자식] 남은요리시간 : 10
2

[부모] 치킨주문을 전달합니다.
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[자식] 치킨주문 도착

[자식] 남은요리시간 : 6
[자식] 남은요리시간 : 8
[자식] 남은요리시간 : 15

[자식] 현재 진행중 요리 남은시간 : 2

[자식] 현재 진행중 요리 남은시간 : 9

[부모] 요리를 전달받아 손님에게 제공하였습니다.
[부모] 현재 진행중인 주문 : 2
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[자식] 현재 진행중 요리 남은시간 : 7

[부모] 요리를 전달받아 손님에게 제공하였습니다.
[부모] 현재 진행중인 주문 : 1
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[부모] 요리를 전달받아 손님에게 제공하였습니다.
[부모] 현재 진행중인 주문 : 0
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
```

4. 모든 주문 완료 후 ctrl-c받았을 때 자식종료후 자신종료

```
[khm970514@lily ch7]$ ./counter.exe
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) : 1

[부모] 피자주문을 전달합니다.
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[자식] 피자주문 도착

[자식] 남은요리시간 : 10
1

[부모] 피자주문을 전달합니다.
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[자식] 피자주문 도착

[자식] 남은요리시간 : 8
[자식] 남은요리시간 : 10
2

[부모] 치킨주문을 전달합니다.
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[자식] 치킨주문 도착

[자식] 남은요리시간 : 6
[자식] 남은요리시간 : 8
[자식] 남은요리시간 : 15

[자식] 현재 진행중 요리 남은시간 : 2

[자식] 현재 진행중 요리 남은시간 : 9

[부모] 요리를 전달받아 손님에게 제공하였습니다.
[부모] 현재 진행중인 주문 : 2
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[자식] 현재 진행중 요리 남은시간 : 7

[부모] 요리를 전달받아 손님에게 제공하였습니다.
[부모] 현재 진행중인 주문 : 1
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[부모] 요리를 전달받아 손님에게 제공하였습니다.
[부모] 현재 진행중인 주문 : 0
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) : ^C
[부모] 자식프로세스를 종료하였습니다.
[khm970514@lily ch7]$
```

5. 조리완료전 주문 3개이상 받고 완료전까지 ctrl-c무시

```
[khm970514@lily ch7]$ ./counter.exe
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) : 1

[부모] 피자주문을 전달합니다.
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[자식] 피자주문 도착

[자식] 남은요리시간 : 10
^C^C^C^C^C^C^C^C^C^C^C^C1

[부모] 피자주문을 전달합니다.
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[자식] 피자주문 도착

[자식] 남은요리시간 : 7
[자식] 남은요리시간 : 10
^C^C^C^C^C^C^C^C^C^C^C^C^C^C^C^C^C^C
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) : 2

[부모] 치킨주문을 전달합니다.
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[자식] 치킨주문 도착

[자식] 남은요리시간 : 4
[자식] 남은요리시간 : 7
[자식] 남은요리시간 : 15
^C^C^C^C^C^C^C^C^C^C^C^C^C^C^C^C^C^C^C^C^C^C^C^C^C
[자식] 현재 진행중 요리 남은시간 : 3

[자식] 현재 진행중 요리 남은시간 : 11

[부모] 요리를 전달받아 손님에게 제공하였습니다.
[부모] 현재 진행중인 주문 : 2
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[자식] 현재 진행중 요리 남은시간 : 8

[부모] 요리를 전달받아 손님에게 제공하였습니다.
[부모] 현재 진행중인 주문 : 1
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[부모] 요리를 전달받아 손님에게 제공하였습니다.
[부모] 현재 진행중인 주문 : 0
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) : ^C
[부모] 자식프로세스를 종료하였습니다.
[khm970514@lily ch7]$ █
```

6. 주문받았을 때 리스트에 추가하고 현재조리중 요리 시간출력,
7. SIGALRM받았을 때 리스트에서 제거하고 현재조리중인 요리 시간 출력 / 통보

```
[khm970514@lily ch7]$ ./counter.exe
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) : 1

[부모] 피자주문을 전달합니다.
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[자식] 피자주문 도착

[자식] 남은요리시간 : 10
1

[부모] 피자주문을 전달합니다.
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[자식] 피자주문 도착

[자식] 남은요리시간 : 8
[자식] 남은요리시간 : 10
2

[부모] 치킨주문을 전달합니다.
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[자식] 치킨주문 도착

[자식] 남은요리시간 : 6
[자식] 남은요리시간 : 8
[자식] 남은요리시간 : 15

[자식] 현재 진행중 요리 남은시간 : 2

[자식] 현재 진행중 요리 남은시간 : 9

[부모] 요리를 전달받아 손님에게 제공하였습니다.
[부모] 현재 진행중인 주문 : 2
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[자식] 현재 진행중 요리 남은시간 : 7

[부모] 요리를 전달받아 손님에게 제공하였습니다.
[부모] 현재 진행중인 주문 : 1
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) :
[부모] 요리를 전달받아 손님에게 제공하였습니다.
[부모] 현재 진행중인 주문 : 0
[부모] 음식을 주문하시겠습니까?
1.피자 2.치킨 (숫자입력, 종료 0) : ^C
[부모] 자식프로세스를 종료하였습니다.
[khm970514@lily ch7]$
```