Unix 프로그래밍 및 실습

7장. 시그널

강의 내용

- 1절 개요
- 2절 시그널의 개념
- 3절 시그널 보내기
- 4절 시그널 기본 처리
- 5절 시그널 집합
- 6절 sigaction 함수의 활용
- 7절 알람 시그널
- 8절 기타 시그널 처리 함수
- http://lily.mmu.ac.kr/lecture/13u2/ch07.pdf
- 책에 나온 내용 반드시 man으로 확인할 것!
  - UNIX, LINUX 등 시스템마다 차이가 있을 수 있음을 반드시 인식

과제(1)

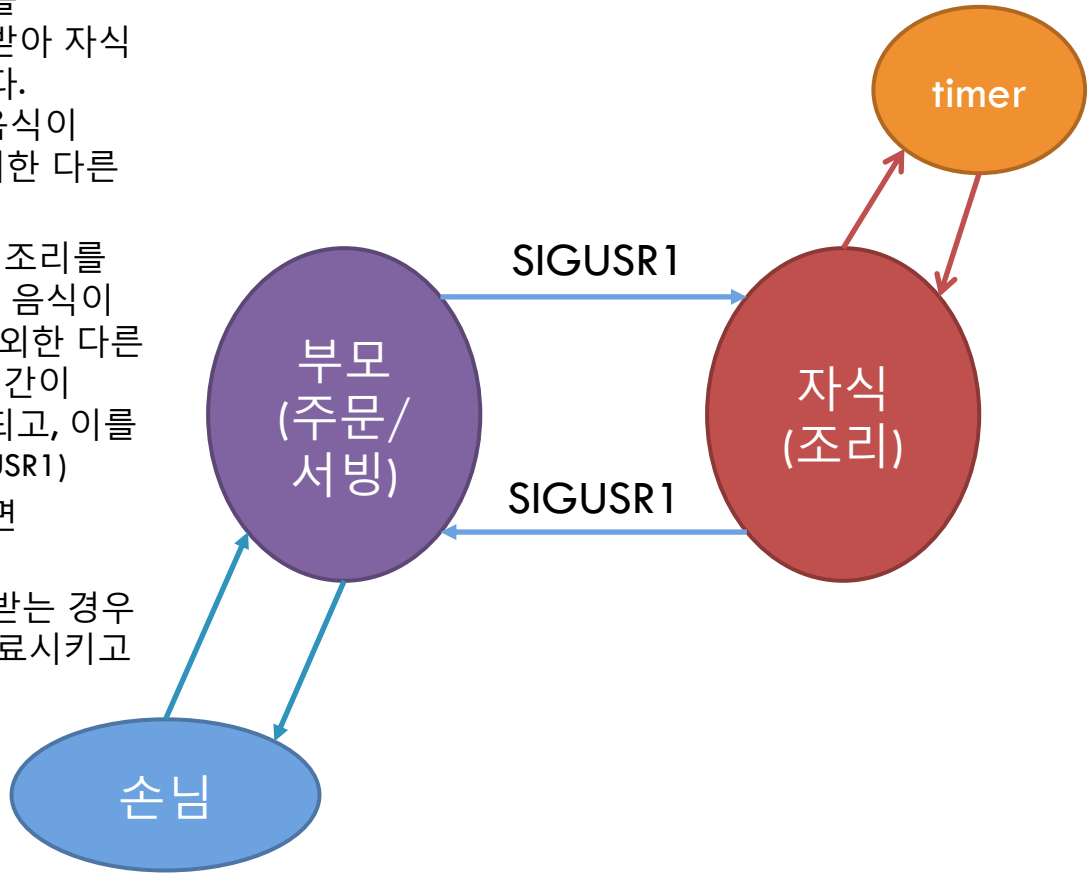
- [예제 7-1] (10점)
  - 과제 개요 (2줄 이상)
  - 프로그램
  - 실행화면 캡처
- [예제 7-2], [예제 7-3], [예제 7-4] (각 10점)
- [예제 7-5] (10점)
- [예제 7-6], [예제 7-7] (각 10점)
- [예제 7-8] (10점)
- [예제 7-9], [예제 7-10] (각 10점)
- [예제 7-11], [예제 7-12], [예제 7-13] (각 10점)
- 보고서 첫 장에 다음 표 작성

과제	완성 여부 (O, X)	비고
예제 7-1 (10)	O	
예제 7-2 (10)	O	
…	O	
예제 7-13 (10)	X	

- 제출기한 : 11월 11일 자정

과제 (2)

- 응용 #1 (1000점)
  - 부모 프로세스는 반복해서 메뉴를 출력하고 사용자로부터 주문을 받아 자식 프로세스에게 주문 내용을 알린다. (SIGUSR1) (일단 주문을 받으면 음식이 완료되기 전까지 SIGUSR1을 제외한 다른 시그널은 모두 무시)
  - 자식 프로세스는 주문을 받으면 조리를 시작한다. (일단 조리를 시작하면 음식이 완성되기 전까지 SIGALARM을 제외한 다른 시그널은 모두 무시) 일정 조리시간이 지내면(Alarm 이용) 음식이 완성되고, 이를 부모 프로세스에게 알린다. (SIGUSR1)
  - 부모 프로세스는 음식이 완성되면 손님에게 알린다. (메시지 출력)
  - 주문이 없는 상태에서 SIGINT를 받는 경우 메시지 출력하고 자식을 먼저 종료시키고 부모 종료



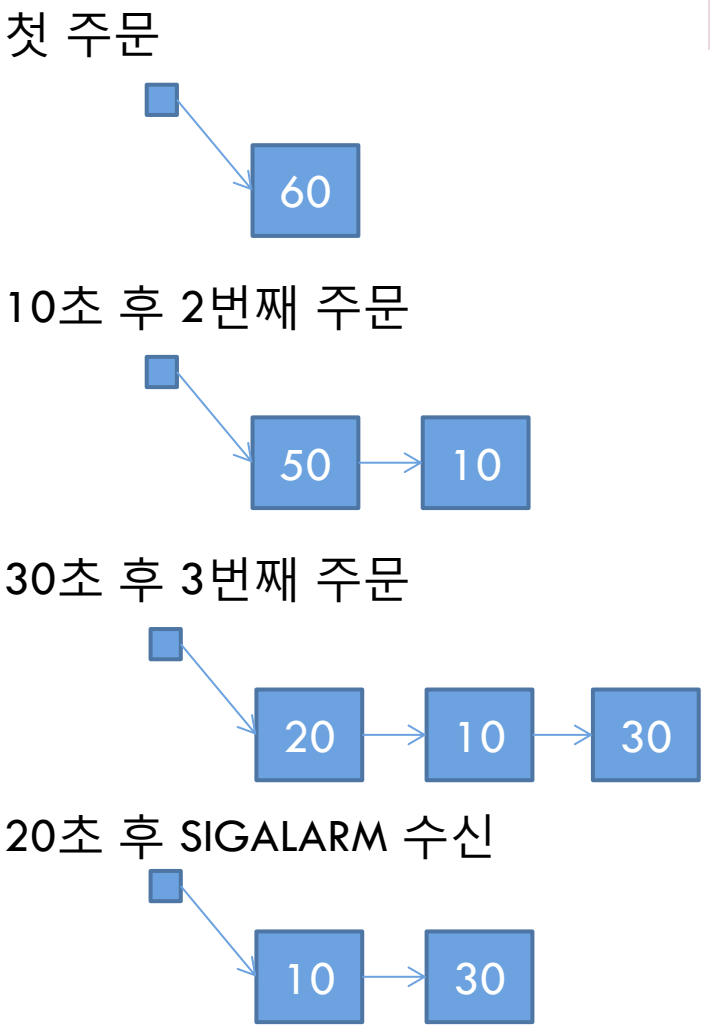
과제 (3)

- 다음 확인표 작성

확인사항	완성 여부 (O, X)	비고
1. (부모) 자식 프로세스 생성 및 exec 정상 작동 여부		
2. (부모) 메뉴 출력 및 사용자로부터 입력 받기		
3. (부모) 주문 받기 전 Ctrl-C 받았을 때 정상 동작(메시지 출력 / 자식 프로세스 종료 / 자신 종료) 여부		
4. (부모) 주문 받은 후 관련 시그널 처리 준비한 후 자식에게 SIGUSR1 보내기		
5, (부모) 조리가 시작된 후 다른 시그널 무시		
6. (자식) 시그널 처리 준비		
7. (자식) SIGUSR1 받았을 때 관련 시그널 처리 준비한 후 조리 시작 (화면에 메시지 출력)		
8. (자식) SIGALARM 받았을 때 관련 시그널 처리 준비한 후 부모에게 통보		
9. (부모) 자식으로부터 SIGUSR1 받았을 때 관련 시그널 처리 준비한 후 손님에게 메시지 출력		
10. (부모) 음식이 완료된 후 Ctrl-C 받았을 때 정상 동작(메시지 출력 / 자식 프로세스 종료 / 자신 종료) 여부		

과제 (4)

- 응용 #2 (1000점)
  - 응용 #1의 경우 자식 프로세스는 한번 조리가 시작되면 다음 주문을 못 받는 상황임.
  - 조리가 시작된 후에도 주문을 계속 받아 조리를 할 수 있도록 개선하시오.
  - 힌트
    - SIGALARM을 받기 전에 alarm(0) 을 하는 경우 남은 시간(초 단위)이 반환됨.
    - Linked list를 이용하여 각 요리의 남은 시간들을 관리



과제 1 예시

- 부모
  - fork
  - while 루프
    - SIGINT 들어오면 자식도 종료하도록 핸들러 등록
    - 화면에 주문 안내 출력
    - 주문이 들어오면 signal 처리함수들 등록
      - SIGUSR1 핸들러 등록
      - SIGINT 무시하도록 설정
  - SIGUSR1 핸들러
    - SIGUSR1이 들어오면 결과 출력
    - SIGINT 핸들러 설정
  - SIGINT 핸들러
    - 자식에게 SIGINT 전달
    - wait 후 메시지 출력
- 자식
  - exec로 몸체 바꾸기
  - signal 처리함수들 등록
    - SIGUSR1 핸들러 등록
  - SIGUSR1 핸들러
    - SIGALRM 핸들러 등록
    - SIGINT 무시하도록 설정
    - 조리시간에 맞추어 alarm 호출
  - SIGALRM 핸들러
    - SIGUSR1 보내기
    - SIGINT 허용하도록 설정

부모 프로세스 (1)

```
 1 #include <sys/types.h>
 2 #include <unistd.h>
 3 #include <signal.h>
 4 #include <stdio.h>
 5 #include <stdlib.h>
 6
 7 int child_pid = -1;
 8
 9 void sigint_handler(int signo)
10 {
11         kill(child_pid, SIGINT);
12         wait();
13         printf("자식 종료\n");
14         exit(0);
15 }
16
17 void sigusr1_handler(int signo)
18 {
19         printf("요리 완료\n");
20         signal(SIGUSR1, SIG_DFL);
21         signal(SIGINT, sigint_handler);
22 }
23
```

부모 프로세스 (2)

```
24 int main(void)
25 {
26         char    ch;
27
28         child_pid = fork();
29
30         if (child_pid < 0) {
31                 perror("fork");
32                 exit(1);
33         }
34         else if (child_pid == 0) {
35                 if (execl("./cook.out", "cook.out", NULL) < 0) {
36                         perror ("execl");
37                         exit(2);
38                 }
39         }
```

부모 프로세스 (3)

```
40          else {
41                  signal(SIGINT, sigint_handler);
42
43                  while (1) {
44                          printf("주문하시겠습니까? (y)");
45                          if (scanf("%c", &ch) == 1) {
46                                  if (ch == 'y') {
47                                          signal(SIGINT, SIG_IGN);
48                                          signal(SIGUSR1, sigusr1_handler);
49                                          kill (child_pid, SIGUSR1);
50                                          printf("주문 전달\n");
51                                          pause();
52                                  }
53                          }
54                  }
55          }
56  }
```

자식 프로세스

```
 1 #include <sys/types.h>
 2 #include <unistd.h>
 3 #include <signal.h>
 4 #include <stdio.h>
 5 #include <stdlib.h>
 6
 7 #define COOK_TIME       10
 8
 9 void sigusr1_handler(int signo);
10 void sigalrm_handler(int signo);
11
12 void sigusr1_handler(int signo)
13 {
14         printf("child: 주문 도착\n");
15         signal(SIGUSR1, SIG_IGN);
16         signal(SIGINT, SIG_IGN);
17         signal(SIGALRM, sigalrm_handler);
18         alarm(COOK_TIME);
19 }
20
```

```
21 void sigalrm_handler(int signo)
22 {
23         printf("child: 요리 완료\n");
24         kill (getppid(), SIGUSR1);
25         signal(SIGINT, SIG_DFL);
26         signal(SIGALRM, SIG_DFL);
27         signal(SIGUSR1, sigusr1_handler);
28 }
29
30 int main(void)
31 {
32         signal(SIGUSR1, sigusr1_handler);
33
34         while (1) {
35                 sleep(60);
36         }
37 }
38
```

과제 2 예시

- 부모
  - fork
  - while 루프
    - SIGINT 들어오면 자식도 종료하도록 핸들러 등록
    - 화면에 주문 안내 출력
    - 주문이 들어오면 이미 조리 중인 음식 수 확인
    - 기존에 조리 중인 음식이 없는 경우
      - SIGUSR1 핸들러 등록
      - SIGINT 무시하도록 설정
    - 조리 중 음식 수 1 증가
    - SIGUSR1 보내기
  - SIGINT 핸들러
    - SIGUSR1이 들어오면 결과 출력
    - 조리 중 음식 수 감소
    - 조리 중인 음식이 없는 경우 SIGINT 핸들러 설정, SIGUSR1 핸들러는 기본 동작
  - SIGINT 핸들러
    - 자식에게 SIGINT 전달
    - wait 후 메시지 출력
- 자식
  - exec로 몸체 바꾸기
  - signal 처리함수들 등록
    - SIGUSR1 핸들러 등록
  - SIGUSR1 핸들러
    - SIGALRM 핸들러 등록
    - SIGINT 무시하도록 설정
    - 조리시간에 맞추어 alarm 호출
  - SIGALRM 핸들러
    - SIGUSR1 보내기
    - SIGINT 허용하도록 설정

부모 프로세스 (1)

```
 1 #include <sys/types.h>
 2 #include <unistd.h>
 3 #include <signal.h>
 4 #include <stdio.h>
 5 #include <stdlib.h>
 6
 7 int child_pid = -1;
 8 int active_no = 0;
 9
10 void sigint_handler(int signo)
11 {
12         kill(child_pid, SIGINT);
13         wait();
14         printf("자식 종료\n");
15         exit(0);
16 }
17
18 void sigusr1_handler(int signo)
19 {
20         printf("요리 완료\n");
21         active_no--;
22         fprintf(stderr, "active : %d\n", active_no);
23         if (active_no == 0) {
24                 sigset(SIGINT, sigint_handler);
25                 sigset(SIGUSR1, SIG_DFL);
26         }
27 }
28
```

부모 프로세스 (2)

```
29 int main(void)
30 {
31         char    ch;
32
33         child_pid = fork();
34
35         if (child_pid < 0) {
36                 perror("fork");
37                 exit(1);
38         }
39         else if (child_pid == 0) {
40                 if (execl("./cook.out", "cook.out", NULL) < 0) {
41                         perror ("execl");
42                         exit(2);
43                 }
44         }
```

부모 프로세스 (3)

```
45          else {
46                  sigset(SIGINT, sigint_handler);
47
48                  while (1) {
49                          printf("주문하시겠습니까? (y)");
50                          if (scanf("%c", &ch) == 1) {
51                                  if (ch == 'y') {
52                                          if (active_no == 0) {
53                                                  sigset(SIGINT, SIG_IGN);
54                                                  sigset(SIGUSR1, sigusr1_handler);
55                                          }
56                                          active_no++;
57                                          fprintf(stderr, "active : %d\n", active_no);
58                                          kill (child_pid, SIGUSR1);
59                                          printf("주문 전달\n");
60                                  }
61                          }
62                  }
63          }
64  }
```

자식 프로세스 (1)

```
 1 #include <sys/types.h>
 2 #include <unistd.h>
 3 #include <signal.h>
 4 #include <stdio.h>
 5 #include <stdlib.h>
 6
 7 #define COOK_TIME       10
 8
 9 typedef struct remaining_list {
10         int secs;
11         struct remaining_list *next;
12 } remaining_list_t;
13
14 remaining_list_t        *header = NULL;
15
16 void sigusr1_handler(int signo);
17 void sigalrm_handler(int signo);
18
```

자식 프로세스 (2)

```
19 void sigusr1_handler(int signo)
20 {
21         remaining_list_t *tmp;
22         remaining_list_t *prev;
23         remaining_list_t *new_el;
24
25         printf("child: 주문 도착\n");
26
27         new_el = malloc(sizeof(remaining_list_t));
28         new_el->secs = COOK_TIME;
29         new_el->next = NULL;
30
31         if (header == NULL) {
32                 sigset(SIGINT, SIG_IGN);
33                 sigset(SIGALRM, sigalrm_handler);
34                 header = new_el;
35         }
36         else {
37                 header->secs = alarm(0);
38
39                 for (tmp = header; tmp; prev = tmp, tmp = tmp->next) {
40                         new_el->secs -= tmp->secs;
41                 }
42                 prev->next = new_el;
43         }
44         fprintf(stderr, "new secs: %d\n", new_el->secs);
45         fprintf(stderr, "new alarm: %d\n", header->secs);
46         alarm(header->secs);
47 }
48
```

자식 프로세스 (3)

```
49 void sigalrm_handler(int signo)
50 {
51         remaining_list_t        *tmp;
52
53         printf("child: 요리 완료\n");
54
55         tmp = header;
56         if (header->next) {
57                 header = header->next;
58                 alarm(header->secs);
59         }
60         else {
61                 header = NULL;
62                 sigset(SIGINT, SIG_DFL);
63                 sigset(SIGALRM, SIG_IGN);
64         }
65
66         kill (getppid(), SIGUSR1);
67         free(tmp);
68 }
69
70 int main(void)
71 {
72         sigset(SIGUSR1, sigusr1_handler);
73
74         while (1) {
75                 sleep(60);
76         }
77 }
```

과제 2 수정

- 주문이 폭주 하는 경우, 주문 간격이 1초가 안되는 상황이 발생할 수 있고, 이 경우 자식 프로세스 40번째 줄에서 0이 되는 상황이 발생할 수 있음
- 이를 방지하기 위해 0이 되는 경우 1초로 강제 설정하는 식으로 개선

```
19 void sigusr1_handler(int signo)
20 {
       ... 생략 ...

37         else {
38                 header->secs = alarm(0);
39
40                 for (tmp = header; tmp; prev = tmp, tmp = tmp->next) {
41                         new_el->secs -= tmp->secs;
42                 }
43                 if (new_el->secs <= 0)  /* edit for rapid orders */
44                         new_el->secs = 1;
45
46                 prev->next = new_el;
47         }

      ... 생략 ...
```