유닉스 개인과제 10 보고서.

실행 결과 201619460 이성규

```
ubuntu@201619460:~/hw10$ gcc -o hw10 hw10.c
ubuntu@201619460:~/hw10$ ./hw10
키로 사용할 파일의 임의 문자열을 입력해 주세요 : ①
Hello every one! <- 직접입력
Initialized:
Received signal:: User defined signal 1  (1)
Child Process:
Received signal:: User defined signal 2  (2)
Changed: Hello every one! <- 파일의 내용
Parent: Child says 0  <- 자식 종료상태확인
```

제출 파일 목록에 키로 쓰일 파일이 압축 파일에 포함되어 있지 않아, 코드 내에서 임의의 문자열을 입력받아 shmfile이란 이름으로 파일을 제작하도록 코드를 짰습니다.

코드

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

#define BUFSIZE (80)

void handler(int signo)
{
    psignal(signo, "Received signal:");
}

int main(void)
{
    pid_t pid;
    int wstatus;
    struct sigaction act;
    key_t key;
    int shmid, size;
    char buf[BUFSIZE];
    void *shmaddr;
    sigset_t set_to_wait_and_block;
    FILE *fp;

    sigemptyset(&act.sa_mask);
    sigaddset(&act.sa_mask, SIGUSR1);
    act.sa_flags = 0;
    act.sa_handler = handler;
    sigaction(SIGUSR1, &act, NULL); //SIGUSR1 handler 설정

    sigemptyset(&act.sa_mask);
    sigaddset(&act.sa_mask, SIGUSR2);
    sigaction(SIGUSR2, &act, NULL); //SIGUSR2 handler 설정
```

핸들러 입니다.
교수님이 구현하신 함수와 동일하게 구현하였습니다.

자식 프로세스 생성 관련

핸들러 등록을 위해

공유 메모리 관련

시그널 대기 및 블럭을 위해

파일 관련

시그널에 대해 핸들러 등록을 하였습니다. 핸들러로, 동일한 함수를 사용하고 있습니다.

```
40      pid = fork();  ──→  자식 프로세서가 분기됨.
41
42      if (pid < 0) //에러
43      {
44          perror("Fork Failed");
45          return 1;
46
47      } else if (pid == 0) //자식 프로세스
48      {
49
50          sigfillset(&set_to_wait_and_block);
51          sigdelset(&set_to_wait_and_block, SIGUSR1); //SIGUSR1만 제외된 집합 설정
52
53          sigprocmask(SIG_BLOCK, &set_to_wait_and_block, NULL); // 시그널 집합에 대해 블록(SIGUSR1빼고 블록)
54
55          sigsuspend(&set_to_wait_and_block); // 집합에서 제외된 시그널(SIGUSR1) 대기
56
                                                SIGUSR1을 받고 핸들러가 호출.(1)
57
58          key = ftok("shmfile",1); //shmfile을 이용해 키 생성. 파일이 존재해야함
59          size = sysconf(_SC_PAGESIZE); //페이지 사이즈 = 4KB
60          shmid = shmget(key, size, IPC_CREAT|0666); //공유 메모리 접근 권한 설정
62          shmaddr = shmat(shmid,NULL,0); //할당 받은 공유 메모리 공간의 주소를 얻음
63
64          memcpy(buf, shmaddr, BUFSIZE); //현재 메모리 공간의 처음 80B 내용 확인.
65          printf("Child Process: %s\n",buf); //0으로 초기화 하였으니 아무런 내용 없음
66
67          fp = fopen("shmfile","r"); //파일오픈
68          if (fp == NULL)
69          {
70              perror("File Open");
71              exit(1);
72          }
                                    파일 안열렸을시 예외처리
            fgets(buf,BUFSIZE,fp);//buf에 파일의 문자열 가져옴
            fclose(fp);
75          memcpy(shmaddr, buf, BUFSIZE); 파일의 문자열인 buf의 문자열을 공유 메모리 공간으로 복사
76
77          shmdt(shmaddr); //공유 메모리 공간 연결 해제
78
79          kill(getppid(),SIGUSR2); 부모 프로세스에게 SIGUSR2 시그널 (2)
80
81          return 0;
83      }else //부모 프로세스
84      {
85          // 키로 사용할 파일은 임의의 문자열로 작성 구현
86          printf("키로 사용할 파일의 임의 문자열을 입력해 주세요 :\n");
87          scanf("%[^\n]s",buf);
88          fp = fopen("shmfile","w");
89          if (fp == NULL)
90          {
91              perror("File Open");
92              exit(1);
93          }
                                    파일 안열렸을시
                                    예외처리
94          fprintf(fp,"%s",buf);
95          fclose(fp);                                  키로 사용할 파일을
96                                                       임의의 문자열로 작성 ①
97          key = ftok("shmfile",1); //shmfile을 이용해 키 생성. 파일이 존재해야함, 앞에서 생성.
98          size = sysconf(_SC_PAGESIZE); //페이지 사이즈 = 4KB
99          shmid = shmget(key, size, IPC_CREAT|0666); //공유 메모리 접근 권한 설정
100
101         shmaddr = shmat(shmid,NULL,0); //할당 받은 공유 메모리 공간의 주소를 얻음
102
103         memset(shmaddr,0,size); //공유 메모리 공간 전체 0으로 초기화
```

```
104
105         memcpy(buf, shmaddr, BUFSIZE); //해당 공간의 처음 80B 만큼의 데이터 복사하여 출력
106         printf("Initialized: %s\n", buf); //0으로 초기화 하였으니 아무내용 없음
107
108         kill(pid,SIGUSR1); // child process에게 시그널 보냄 자식 프로세스에게 SIGUSR1 시그널 (1)
109
110         sigfillset(&set_to_wait_and_block);
111         sigdelset(&set_to_wait_and_block, SIGUSR2); //SIGUSR2만 제외된 집합 설정
112
113         sigprocmask(SIG_BLOCK, &set_to_wait_and_block, NULL); // 시그널 집합에 대해 블록(SIGUSR2빼고 블록)
114
115         sigsuspend(&set_to_wait_and_block); // 집합에서 제외된 시그널(SIGUSR2) 대기
116         SIGUSR2를 받고 핸들러가 호출(2)
117         //sigprocmask(SIG_UNBLOCK,&set_to_wait_and_block, NULL); //블록해제
118
119
120         memcpy(buf, shmaddr, BUFSIZE); //해당 공간의 변경된 내용 확인
121         printf("Changed: %s\n", buf);
122
123         shmdt(shmaddr);
124         shmctl(shmid, IPC_RMID,NULL); //공유 메모리 공간 할당 해제
125
126
127
128         wait(&wstatus);                    자식 프로세스 종료 상태 확인
129         printf("Parent: Child says %d\n",wstatus);
130
131         return 0;
132     }
```

블록 해제를 해야하나 코드를 구현했으나, 딱히 과제에 명시되어 있지않아 주석을 풀지 않음.

코드 관련하여:

1. 29~37 라인 : struct sigaction를 같은 데이터로 써도 딱히 문제가 없는지 몰랐어서 따로 코드를 만들어서 실행 해보았습니다. 교수님이 만드신 코드와 동일하게 만들었습니다.

이는 딱히 상관이 없었고, 둘 다 핸들러 등록이 잘되었습니다.

```
13  int main(void)
14  {
15      int i;
16      struct sigaction act;
17
18      sigemptyset(&act.sa_mask);
19      sigaddset(&act.sa_mask, SIGUSR1);
20      act.sa_flags = 0;
21      act.sa_handler = handler;
22      sigaction(SIGUSR1, &act, NULL); //handler 설정
23
24      sigemptyset(&act.sa_mask);
25      sigaddset(&act.sa_mask, SIGUSR2);
26      sigaction(SIGUSR2, &act, NULL); //handler 설정
27
28      for(i=0; i<50; i++)
29      {
30          printf("waiting for signal %d or %d : %02d seconds\n",SIGUSR1,SIGUSR2,i);
31          sleep(1);
32      }
33
34      printf("Finish!\n");
35
36
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Received signal:: User defined signal 2
waiting for signal 10 or 12 : 41 seconds
Received signal:: User defined signal 2
waiting for signal 10 or 12 : 42 seconds
waiting for signal 10 or 12 : 43 seconds
waiting for signal 10 or 12 : 44 seconds
Received signal:: User defined signal 1
waiting for signal 10 or 12 : 45 seconds

Sending signal 12 to 171205
ubuntu@201619460:~/hw10$ ./s1 10 171205
Sending signal 10 to 171205
ubuntu@201619460:~/hw10$ ./s1 10 171205
Sending signal 10 to 171205
ubuntu@201619460:~/hw10$ ./s1 10 171205
Sending signal 10 to 171205
ubuntu@201619460:~/hw10$
```

2. 86~95 라인(①) : 키로 사용할 파일을 만드는 부분으로, 제출한 코드에는 부모 프로세스에 구현이 되어있는데, 처음에는 fork함수가 호출되기 전에 구현을 하려 했습니다. 그러나 이상하게도 코드가 제대로 실행이 되지 않고 멈추는 현상이 일어나 많이 고생을 하였습니다.

```
40      // 이 부분 키로 사용할 파일은 임의의 문자열로 작성 구현
41      printf("키로 사용할 파일의 임의 문자열을 입력해 주세요 : \n");
42      scanf("%[^\n]s",buf);
43      fp = fopen("shmfile","w");
44      if (fp == NULL)
45      {
46          perror("File Open");
47          exit(1);
48      }
49      fprintf(fp,"%s",buf);
50      fclose(fp);
51
52
53
54      printf("Parent: mypid = %d Fork!\n", getpid());
55
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

ubuntu@201619460:~/hw10$
ubuntu@201619460:~/hw10$
ubuntu@201619460:~/hw10$
ubuntu@201619460:~/hw10$ gcc -o hw10 hw10.c
ubuntu@201619460:~/hw10$ ./hw10
키로 사용할 파일의 임의 문자열을 입력해 주세요 :
memory if i know
Parent: mypid = 449478 Fork!
Parent process's Shared memory info: key=16906113 shmid=2 shmaddr=0x7f5ba1004000
Initialized:
Received signal:: User defined signal 1
```

무엇이 문제인지 파악하고자 fclose(fp) 함수를 지우거나 %[^\n]s 형식을 바꾸는 등 여러 시도를 하였으나, fork함수 호출 전에 구현하면 오류가 생겼습니다.

```
} else if (pid == 0) //자식 프로세스
{
    printf("child process start");

    sigfillset(&set_to_wait_and_block);
    printf("child process start");
    sigdelset(&set_to_wait_and_block, SIGUSR1); //SIGUSR1만 제외된 집합 설정
    printf("child process start");

    sigprocmask(SIG_BLOCK, &set_to_wait_and_block, NULL); // 시그널 집합에 대해 블록(SIGUSR1빼고 블록)
    printf("child process start");

    sigsuspend(&set_to_wait_and_block); // 집합에서 제외된 시그널(SIGUSR1) 대기
    printf("child process start");


    key = ftok("shmfile",1); //shmfile을 이용해 키 생성. 파일이 존재해야함
    size = sysconf(_SC_PAGESIZE); //페이지 사이즈 = 4KB
    shmid = shmget(key, size, IPC_CREAT|0666); //공유 메모리 접근 권한 설정
```

```
// 이 부분 키로 사용할 파일은 임의의 문자열로 작성 구현
fp = fopen("shmfile","w");
if (fp == NULL)
{
    perror("File Open");
    exit(1);
}
fprintf(fp,"%s","randsdsadasd asdasda asd");
fclose(fp);
```

```
ubuntu@201619460:~/hw10$ gcc -o hw10 hw10.c
ubuntu@201619460:~/hw10$ ./hw10
Parent: mypid = 481782 Fork!
Parent process's Shared memory info: key=16906113 shmid=6 shmaddr=0x7febcf3fa000
Initialized:
Received signal:: User defined signal 1
```

이는 코드가 자체의 오류가 아니라 fork함수 이전에 이 코드를 구현하게 되면 자식 프로세스와 부모 프로세스에서 둘 다 코드가 수행되며 shmfile 파일을 중복되게 같이 열게 되고, 그로인해 오류가 생기는 것이라고 판단하였습니다.