LAB06
시스템 프로그래밍_류은경**Prof**

2021115744_권구태

1. play_again3.c 수정

- SIGINT, SIGQUIT signal을 무시하고, y 또는 n 값만 입력 받음
- signal handler는 아래와 같이 signum 메시지를 출력

[코드 설명**]**

```
    signal(SIGINT, signal_handler);
    signal(SIGQUIT, signal_handler);
```

기존의 play_again3.c 파일에서 위 의 부분만 추가함. signal 함수는 C언어에서 signal 처리를 설정하는 함수. 첫번째 인자에는 어떤 시그널을 처리할지 (정수형 숫자를 넣어줌). 두번째 인자에는 해당 함수가 실행되었을때 실행될 handler 함수. 내가 만든 함수를 넣어도 되고, 기존에 signal 라이브러리에서 제공하는 함수를 사용해도 된다.

```
void signal_handler(int signum) {
    signalHandlerFlag = 1;
    printf("\nSignal %d recieved, but ignored it.\n", signum);
}
```

함수를 추가로 작성하여 signal 함수의 두번째 인자에 넣어준다. signal 함수가 실행되면, 아래의 printf문이 실행됨. signal_handler함수는 handler 함수로써, signal 함수가 실행될때 SIGINT == 2, SIGQUIT == 3이 자동으로 파라미터로 들어감. 그리고 flag변수를 1로 지정해줌. flag 변수는 전역변수.

```
    while (1) {
        tty_mode(0);                    // Save terminal settings
        set_cr_noecho_mode();           // Set -icanon, -echo
        set_nodelay_mode();             // Set non-blocking
        signal(SIGINT, signal_handler); // Register signal
handlers
        signal(SIGQUIT, signal_handler);

        signalHandlerFlag = 0;
        response = get_response(ASK, TRIES);

        tty_mode(1);                    // Restore terminal
settings
```

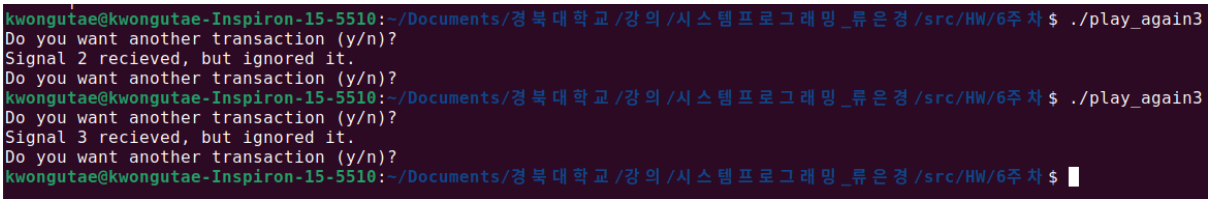
```
        if (response == 3) {
            continue; // restart main loop
        }

        return response; // normal exit
    }
```

만약 response가 3이라면, main함수를 다시 시작.다시 질문을 받음.

```
if (signalHandlerFlag == 1) return 3;
```

get_response 함수 안의 while문 안의 가장 위에 해당 구문을 추가하여 signal_handler함수가 실행되었다면 3을 반환하고, 이는 메인 함수의 while문이 다시 실행되도록 할 수 있음.

[실행결과]



[주의할 점 및 새롭게 알게된 점]

1) signal함수에 두번째 인자에 들어가는 핸들러 함수의 파라미터에 signal 함수의 첫번째 인자의 int형 정수가 들어간다는 점.
2) signal 함수의 첫번째 인자에 들어가는 것은 모두 정수
3) stty 0, 1 따로 설정해서(if 문으로) 백업 및 복원 작업 해줘야 함. 그렇지 않으면 프로그램이 종료되어도 터미널이 정상적으로 돌아오지 않음.
4) fflush(stdout)함수는 현재 버퍼에 저장된 모든 내용을 즉시 화면에 출력한다.
5) 아래는 nodelay_mode()함수 설명
6) fcntl()은 파일 디스크립터의 속성을 조작하는 함수
7) O_NDELAY는 non-blocking 모드를 의미.
8) 비트 OR 연산으로 기존 플래그에 O_NDELAY를 추가.
9) 이러면 read(), getchar() 등에서 입력이 없으면 바로 EOF나 -1을 반환함.
10) F_SETFL로 방금 수정한 플래그를 다시 적용함.
11) 이제 stdin은 non-blocking 입력 대기 모드로 작동함.

2. (rotate.c 수정) 터미널 설정 제어

- canonical 모드 및 echo 기능을 끄고, 입력된 문자에 대해 다음 알파벳을 출력

- 'Q'를 누르면 설정을 복원하고 종료

[코드 설명]

```
    tty_mode(0);                /* restore tty mode */
    set_cr_noecho_mode();           /* set -icanon, -echo   */

    while( ( c=getchar())!= 'Q' ) {
        if( c == 'z' )
            c = 'a';
        else if ( islower(c) )
            c++;
        putchar(c);
    }

    tty_mode(1);
```

기존의 rotate.c 코드에서 tty 함수와 noecho_mode 함수를 추가함.
tty를 복원해야해서 ehco모드 설정 함수 호출 위 아래로 감싼다.
그리고 Q가 들어오면 종료하는 코드 작성.

[실행결과]

```
kwongutae@kwongutae-Inspiron-15-5510:~/Documents/경북대학교/강의/시스템프로그래밍_류은경/src/HW/6주차$ ./rotate
bdgkwongutae@kwongutae-Inspiron-15-5510:~/Documents/경북대학교/강의/시스템프로그래밍_류은경/src/HW/6주차$
```

[주의할 점 및 새롭게 알게된 점]

1. 터미널을 원래의 상태로 되돌리기 위해 tty함수를 echo 및 icanon 작업이 들어간 함수 호출 앞(백업) 뒤(되돌리기)로 호출하여 작업을 해줘야 한다.
2. 에코모드 끄기는 `ttystate.c_lflag &= ~ECHO;` 이렇게 ~ECHO를 사용하여 비트연산을 실행하면 되지만, 에코를 켤때는 ~만 뺀다고 되는게 아님. 그냥 기존의 구조체 자체를(백업 해뒀던) 구조체 자체를 다시 대입해주는게 에러를 일으키지 않을 가능성이 높다.
3. ttystat.c_cc[VMIN] = 1 -> 문자를 하나씩 받겠다는 의미.

3. 소수 탐색 프로그램 작성

- 정수 n이 소수인지 판단하는 함수 작성 int is_prime(unsigned long n); /* n이 소수이면 1, 아니면 0 반환 */
- 2부터 시작하여 정수를 1씩 증가시키며 is_prime() 함수를 사용해 소수 여부 확인: 현재까지 확인한 정수의 개수 cnt와 가장 큰 소수 p를 저장
- 사용자가 Ctrl+C(SIGINT)를 누르면 cnt 와 p를 출력
- SIGINT를 3회 누르면 프로그램이 종료

[코드 설명]

```
int is_prime(unsigned long num) {
   for (unsigned long i = 2; i < num; i++) {
       if(num % i == 0) {
```

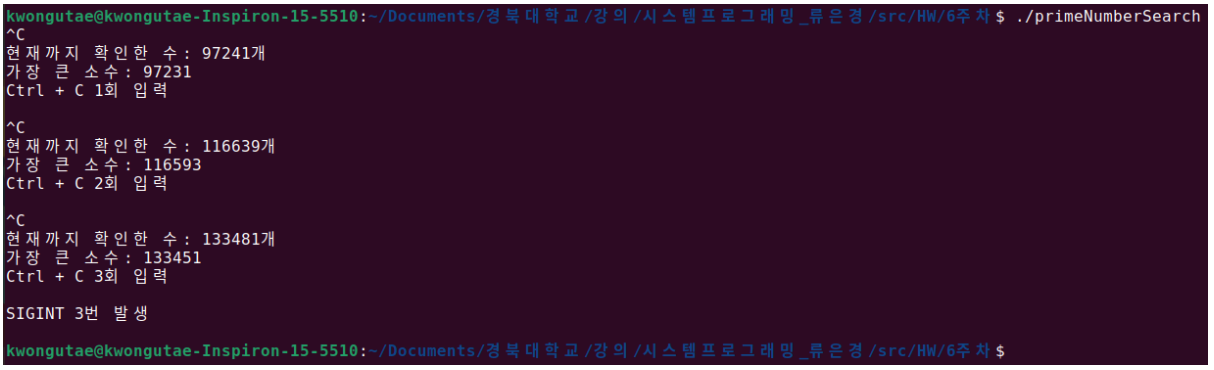
```
            return 0;
        }
    }
    return 1;
}

void signal_handler(int signum) {
    CallSignalCount++;
    printf("\n현재까지 확인한 수: %lu개\n", cnt);
    printf("가장 큰 소수: %lu\n", p);
    printf("Ctrl + C %d회 입력\n\n", CallSignalCount);
    if (CallSignalCount == MAXCOUNT) {
        printf("SIGINT %d번 발생\n\n", MAXCOUNT);
        exit(130);
    }
}
```

signal_handler 함수와 소수인지를 판별하는 is_prime 함수의 코드 내용이다.
세번부르면 종료할 수 있는 exit() 함수도 추가함.

```
    signal(SIGINT, signal_handler);

    while (1)
    {
        int bool = is_prime(cnt);
        if (bool == 1 && cnt != 1) {
            p = cnt;
        }
        cnt++;
    }
```

메인 함수에서는 signal을 받고 이를 처리할 수 있는 signal함수와, 소수를 무한으로 구하는 반복문을 작성함.

[실행 결과]



[주의할 점 및 새롭게 알게된 점]

1) signal함수의 사용법
2) 소수 구하는 것은 무한반복 loop이기 때문에 저장할 변수의 자료형을 int로 설정하면 오버헤드가 날 수 있음. 따라서 unsigned long 으로 지정해줘야 한다.