**운영체제 Homework #1**

소프트웨어학과 2021041017 김규현

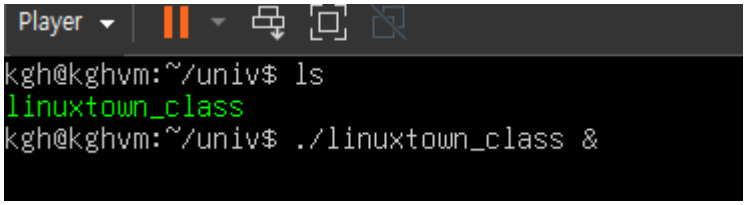
**[고안한 과제 해결 방법]**

1. 일단 Linux 터미널에서 200개 프로세스에 'SIGUSR1' 시그널을 순차적으로 보낸다.

2. 진행하다보면 터미널에 '! !'가 출력되면서 전송한 'SIGUSR1' 시그널을 받았다는 것을 확인할 수 있고, confession.txt에 어떤 문자가 작성되었는지 확인한다.

3. 이제 C언어로 코드를 작성하는데, SIGUSR1 시그널을 받으면 그 문자를 기록하는 프로세스를 찾는 코드를 작성한다.

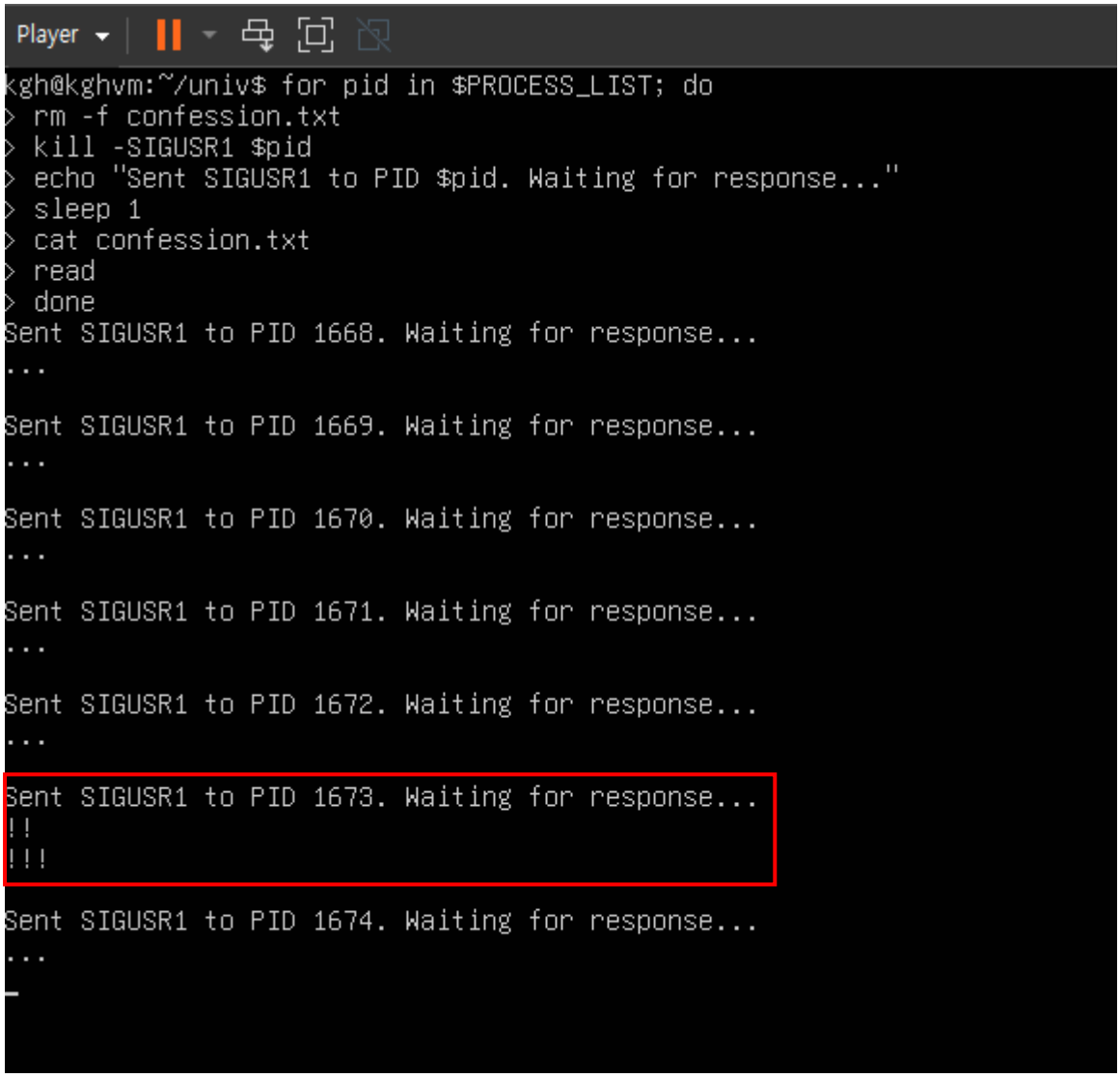
**[추리 과정]**




'linuxtown_class' 파일을 './linuxtown_class &' 명령어로 백그라운드에서 실행한다.

```
kgh@kghvm:~/univ$ echo $!
1668
kgh@kghvm:~/univ$ PARENT=1668
kgh@kghvm:~/univ$ PROCESS_LIST="$PARENT $(ps --no-headers -o pid --ppid $PARENT)"
kgh@kghvm:~/univ$
```

부모 PID를 확인 후 PROCESS_LIST에 저장하고, 자식들을 추가한다.




터미널에 다음과 같은 명령어를 입력한다:

```
for pid in $PROCESS_LIST; do
        rm -f confession.txt      // confession.txt 삭제 (초기화)
        kill -SIGUSR1 $pid        // 해당 프로세스에 SIGUSR1 전송

        // 영어가 안정적이라 영어로 작성
        echo "Sent SIGUSR1 to PID $pid. Waiting for respose..."
        sleep 1       // 프로세스가 기록할 시간을 충분히 준다
        cat confession.txt        // 파일 내용을 확인
        read    // 확인 후 다음 프로세스로 진행
done
```

여기서 터미널을 보면 SIGUSR1 시그널을 받은 프로세스가 시그널 핸들러에 의해 '!!'가 출력되면서 confession.txt 파일에 '!!!'의 문자열을 추가한 모습을 확인할 수 있다.

→ 마피아 프로세스는 confession.txt에 '!!!' 문자열을 추가한다는 점을 알 수 있다.

그럼 이제 C언어 코드를 SIGUSR1 시그널을 받았을 때, '!!!'을 confession.txt에 작성하는 프로세스 목록들을 추리는 로직으로 작성하면 마피아 프로세스를 검거할 수 있다.

**[로직에 맞게 C언어 코드 작성]**

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

#define MAX_CHILD 200

int main(int argc, char *argv[]){
    if (argc < 2) {
        fprintf(stderr, "Please insert Parent PID\n");
        return 1;
    }

    int parentPID = atoi(argv[1]);
    int processPids[MAX_CHILD], processCnt = 0;
    processPids[processCnt++] = parentPID;

    char cmd[128];
    snprintf(cmd, sizeof(cmd), "ps --no-headers -o pid --ppid %d", parentPID);

    FILE *fp = popen(cmd, "r");
    if (!fp) {
        perror("popen");
        return 1;
    }

    char line[256];
    while (fgets(line, sizeof(line), fp)) {
        int pid = atoi(line);
        if (pid > 0 && processCnt < MAX_CHILD) {
            processPids[processCnt++] = pid;
        }
    }
    pclose(fp);

    remove("confession.txt");

    for (int i = 0; i < processCnt; i++) {
        kill(processPids[i], SIGUSR1);
        usleep(300000);
    }

    sleep(1);

    FILE *cf = fopen("confession.txt", "r");
    if (!cf) {
```

(... 코드 이하 생략, 압축 폴더 내에 .c 파일로 있거나 이 문서 끝에 존재한다.)

고안한 로직대로 C언어 코드를 작성하고 저장한다.

**[코드 실행결과]**

```
kgh@kghvm:~/univ$ ls
answer.c  linuxtown_class
kgh@kghvm:~/univ$ ./linuxtown_class &
```

확실한 테스트를 위해 재부팅 후, 'linuxtown_class' 파일을 백그라운드에서 실행한다.

```
kgh@kghvm:~/univ$ echo $!
1149
kgh@kghvm:~/univ$ gcc -o answer answer.c
kgh@kghvm:~/univ$ ./answer 1149
!!
!!
!!
!!
!!
!!
!!
!!
!!
!!
!!
!!
!!
!!
!!
!!
mafia = 17
citizen = 183
=== mafia list ===

1159
1161
1169
1186
1196
1201
1214
1220
1236
1246
1247
1251
1255
1276
1294
1303
1343
kgh@kghvm:~/univ$ _
```

부모 PID 확인 후, gcc로 answer.c 파일을 컴파일 하고 난 뒤에,

부모 PID를 인자로 전달하여 answer 파일을 실행한다.

→ 마피아 프로세스와 시민 프로세스가 잘 추려졌다.

**C언어 코드(압축 파일 내에도 첨부):**

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

#define MAX_CHILD 200

int main(int argc, char *argv[]){
    if (argc < 2) {
        fprintf(stderr, "Please insert Parent PID₩n");
        return 1;
    }

    int parentPID = atoi(argv[1]);     // 인자를 정수형으로 변환하여 부모 PID로 사용.
    int processPids[MAX_CHILD], processCnt = 0;
    processPids[processCnt++] = parentPID;       // 부모 프로세스도 검사 대상에 포함.

    // 부모 PID의 자식 프로세스 목록을 가져오기 위한 ps 명령어 문자열 생성.
    char cmd[128];
    snprintf(cmd, sizeof(cmd), "ps --no-headers -o pid --ppid %d", parentPID);
```

```
FILE *fp = popen(cmd, "r");    // 위 명령어를 실행하여 자식 PID 목록을 읽어옴.
if (!fp) {
    perror("popen");
    return 1;
}

char line[256];
while (fgets(line, sizeof(line), fp)) {
    int pid = atoi(line);  // 문자열을 정수 PID로 변환.
    if (pid > 0 && processCnt < MAX_CHILD) {
        processPids[processCnt++] = pid;  // 유효한 PID이면 배열에 추가.
    }
}
pclose(fp);

// confession.txt 파일 삭제 (초기화)
remove("confession.txt");

// 모든 대상 프로세스들에게 SIGUSR1 전송
for (int i = 0; i < processCnt; i++) {
    kill(processPids[i], SIGUSR1);
    usleep(300000);  // 파일에 기록할 시간 확보
}

// 모든 프로세스가 기록을 완료할 시간을 추가로 대기
sleep(1);
```

```
// confession.txt를 읽어서 결과를 분석
FILE *cf = fopen("confession.txt", "r");
if (!cf) {
    perror("fopen confession.txt");
    return 1;
}

// 각 줄은 해당 프로세스가 기록한 것으로 추리
char *res[MAX_CHILD];
int resCnt = 0;
while (resCnt < processCnt && fgets(line, sizeof(line), cf)) {
    int len = strlen(line);
    if (len > 0 && line[len-1] == '₩n')
        line[len-1] = '₩0';
    res[resCnt++] = strdup(line);
}
fclose(cf);

// "!!!"가 기록된 경우 마피아 프로세스로 간주
int mafiaCnt = 0, mafiaPids[MAX_CHILD];
for (int i = 0; i < resCnt; i++) {
    if (strcmp(res[i], "!!!") == 0) {
        mafiaPids[mafiaCnt++] = processPids[i];
    }
    free(res[i]);
}
```

```
    // 최종 결과 출력
    printf("mafia = %d₩n", mafiaCnt);
    printf("citizen = %d₩n", processCnt - mafiaCnt);
    printf("=== mafia list ===₩n₩n");
    for (int i = 0; i < mafiaCnt; i++) {
        printf("%d₩n", mafiaPids[i]);
    }

    return 0;
}
```

끝.