2018 시스템 프로그래밍
- Lab 08 -

제출일자	2018.11.26
분 반	02
이 름	장수훈
학 번	201402414

Trace 08

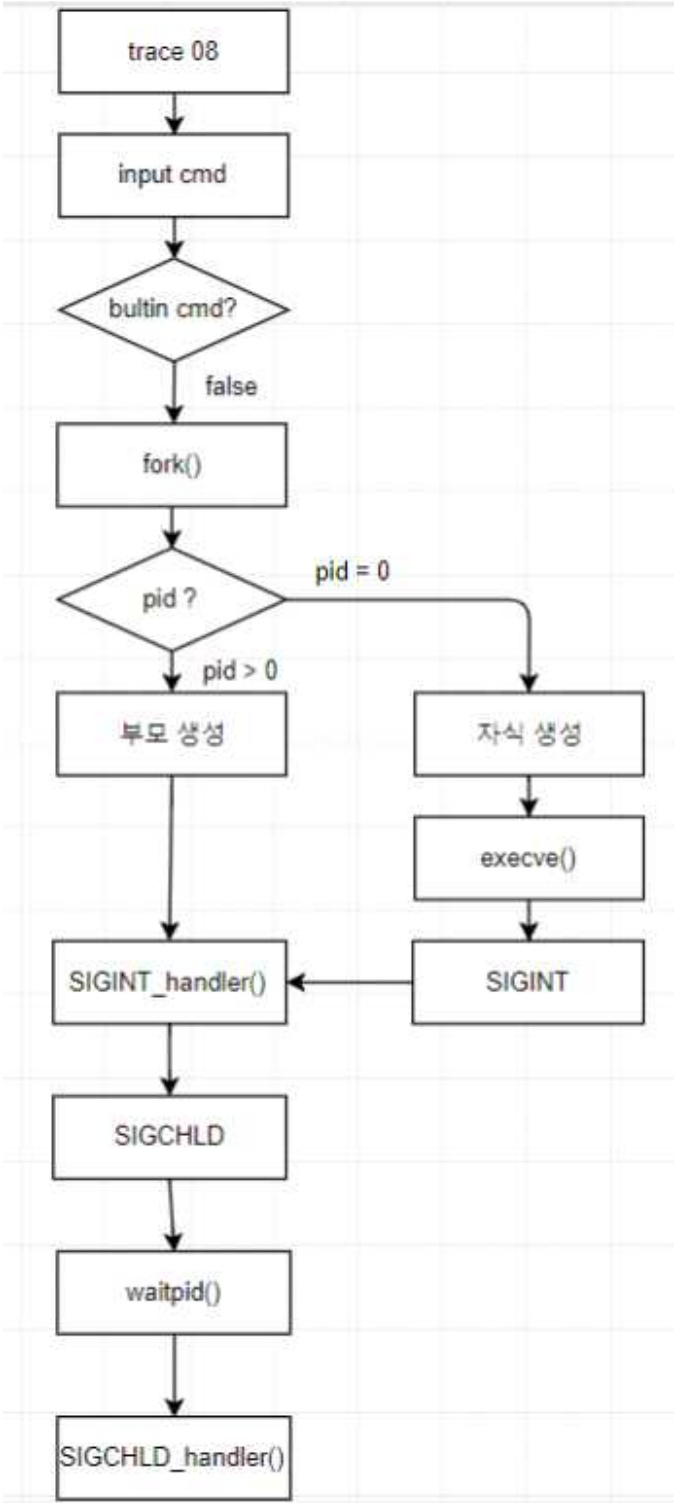
```
c201402414@2018-sp:~/shell/shlab-handout$ ./sdriver -V -t 08 -s ./tshref
Running trace08.txt...
Success: The test and reference outputs for trace08.txt matched!
Test output:
#
# trace08.txt - Send fatal SIGINT to foreground job.
#
tsh> ./myintp
Job [1] (16472) terminated by signal 2
tsh> quit

Reference output:
#
# trace08.txt - Send fatal SIGINT to foreground job.
#
tsh> ./myintp
Job [1] (16480) terminated by signal 2
tsh> quit
```

```
c201402414@2018-sp:~/shell/shlab-handout$ ./sdriver -V -t 08 -s ./tsh
Running trace08.txt...
Success: The test and reference outputs for trace08.txt matched!
Test output:
#
# trace08.txt - Send fatal SIGINT to foreground job.
#
tsh> ./myintp
Job [1] (17065) terminated by signal 2
tsh> quit

Reference output:
#
# trace08.txt - Send fatal SIGINT to foreground job.
#
tsh> ./myintp
Job [1] (17075) terminated by signal 2
tsh> quit
```

각 trace 별 플로우 차트



trace 해결 방법 설명

```
void eval(char *cmdline)
{
    char *argv[MAXARGS];
    pid_t pid;
    int bg;
    sigset_t mask;

    //sigemptyset(&mask); // 초기화
    //sigaddset(&mask,SIGCHLD);// 시그널셋에 추가
    //sigaddset(&mask,SIGINT); // 추가

    bg = parseline(cmdline,argv);
    //명령어를 parseline을 통해 분리

    if (!builtin_cmd(argv)){

        sigemptyset(&mask);
        sigaddset(&mask, SIGCHLD);

        sigprocmask(SIG_BLOCK,&mask,NULL);//시그널 블록

        if((pid = fork())== 0) {

            sigprocmask(SIG_UNBLOCK,&mask,NULL);//시그널 언블록
            setpgid(0,0);
            if((execve(argv[0], argv, environ) < 0 )){
                printf("%s : Command not found\n", argv);
                exit(0);
            }
        }
        else if(pid<0){
            unix_error("fork error");
        }
        addjob(jobs, pid,(bg ==1?BG:FG), cmdline);

        sigprocmask(SIG_UNBLOCK,&mask,NULL);

        if(!bg){
            waitfg(pid,STDOUT_FILENO);
        }
        else{
            printf("(%d) (%d) %s", pid2jid(pid), pid, cmdline);
        }
    }
    // parsing된 명령어를 전달
    //builtin_cmd(argv);

    return;
```

```c
void sigchld_handler(int sig)
{
    pid_t pid;
    int status=0;
    while((pid=waitpid(-1,&status,WNOHANG|WUNTRACED))>0){
        if(WIFSIGNALED(status)){
            printf("Job [%d] (%d) terminated by signal %d\n",pid2jid(pid),pid,WTERMSIG(status));
            deletejob(jobs,pid);
        }
        else if(WIFSTOPPED(status)){
            printf("Job [%d] (%d) stopped by signal %d\n",pid2jid(pid),pid,WSTOPSIG(status));
        getjobpid(jobs,pid)->state=ST;
        }
        else if(WIFEXITED(status)){
            deletejob(jobs,pid);
        }
    }
```

```c
void sigint_handler(int sig)
{
    pid_t pid = fgpid(jobs);
    if(pid!=0){
        kill(-pid,sig);
    }
    return;
    /*
    pid_t pid;
    if((pid=fgpid(jobs))>0){
        kill(pid,SIGINT);
    }
    return;
*/
}
```

trace08은 SIGINT(컨트롤c)가 입력되면 foreground 작업을 kill하는 것이다.
fork를 통해 자식 프로세서를 만들고. execve로 실행파일을 만들고 실행파일을 실행시킬시 부모 프로세스로 SIGINT 시그널을 보낸다. 부모는 시그널을 받았을시 핸들러를 통해서 실행중인 자식 프로세스로 kill함수를 통해 SIGINT 시그널을 보내고, 실행 파일이 종료된다음 부모 프로세서로 SIGCHLD 시그덜을 보낸다. 그뒤 시그널을 받은 부모는 핸들러를 처리하는 것이 이 단계의 과정이라고 할 수 있다.

Trace 09

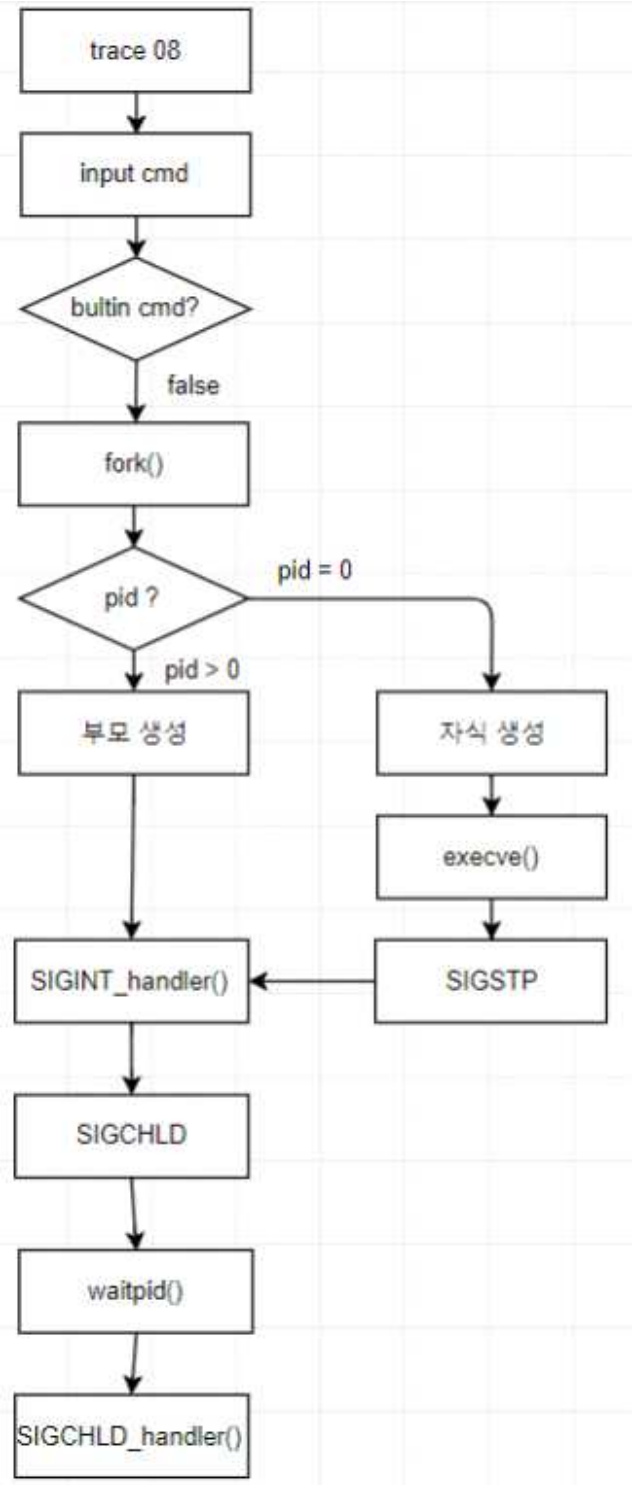
```
c201402414@2018-sp:~/shell/shlab-handout$ ./sdriver -V -t 09 -s ./tshref
Running trace09.txt...
Success: The test and reference outputs for trace09.txt matched!
Test output:
#
# trace09.txt - Send SIGTSTP to foreground job.
#
tsh> ./mytstpp
Job [1] (23224) stopped by signal 20
tsh> jobs
(1) (23224) Stopped ./mytstpp

Reference output:
#
# trace09.txt - Send SIGTSTP to foreground job.
#
tsh> ./mytstpp
Job [1] (23232) stopped by signal 20
tsh> jobs
(1) (23232) Stopped ./mytstpp
```

```
c201402414@2018-sp:~/shell/shlab-handout$ ./sdriver -V -t 09 -s ./tsh
Running trace09.txt...
Success: The test and reference outputs for trace09.txt matched!
Test output:
#
# trace09.txt - Send SIGTSTP to foreground job.
#
tsh> ./mytstpp
Job [1] (23260) stopped by signal 20
tsh> jobs
(1) (23260) Stopped    ./mytstpp

Reference output:
#
# trace09.txt - Send SIGTSTP to foreground job.
#
tsh> ./mytstpp
Job [1] (23268) stopped by signal 20
tsh> jobs
(1) (23268) Stopped ./mytstpp
```

각 trace 별 플로우 차트



trace 해결 방법 설명

[코드는 08과 동일]

trace09는 SIGSTP 실행 시 컨트롤 z를 입력 시 foreground작업을 종료시키는 것이다. 작업중인 프로세서를 종료시키는 시그널을 구현하기 위해선 sigstp_handler를 구현하고 이것 eval 함수에 추가한 것이다. 자식 프로세스의 실행/종료 여부에 따라서 반환해주는 WIFSTOPPED를 사용하여 변경한다.

Trace 10

```
c201402414@2018-sp:~/shell/shlab-handout$ ./sdriver -V -t 10 -s ./tshref
Running trace10.txt...
Success: The test and reference outputs for trace10.txt matched!
Test output:
#
# trace10.txt - Send fatal SIGTERM (15) to a background job.
#
tsh> ./myspin1 5 &
(1) (23907) ./myspin1 5 &
tsh> /bin/kill myspin1
kill: failed to parse argument: 'myspin1'
tsh> quit

Reference output:
#
# trace10.txt - Send fatal SIGTERM (15) to a background job.
#
tsh> ./myspin1 5 &
(1) (23917) ./myspin1 5 &
tsh> /bin/kill myspin1
kill: failed to parse argument: 'myspin1'
tsh> quit
```

```
c201402414@2018-sp:~/shell/shlab-handout$ ./sdriver -V -t 10 -s ./tsh
Running trace10.txt...
Success: The test and reference outputs for trace10.txt matched!
Test output:
#
# trace10.txt - Send fatal SIGTERM (15) to a background job.
#
tsh> ./myspin1 5 &
(1) (23949) ./myspin1 5 &
tsh> /bin/kill myspin1
kill: failed to parse argument: 'myspin1'
tsh> quit

Reference output:
#
# trace10.txt - Send fatal SIGTERM (15) to a background job.
#
tsh> ./myspin1 5 &
(1) (23959) ./myspin1 5 &
tsh> /bin/kill myspin1
kill: failed to parse argument: 'myspin1'
tsh> quit
```

각 trace 별 플로우 차트

[trace 08 09와 동일]

trace 해결 방법 설명

```
void sigchld_handler(int sig)
{
    pid_t pid;
    int status=0;
    while((pid=waitpid(-1,&status,WNOHANG|WUNTRACED))>0){
        if(WIFSIGNALED(status)){
            printf("Job [%d] (%d) terminated by signal %d\n",pid2jid(pid),pid,WTERMSIG(status));
            deletejob(jobs,pid);
        }
        else if(WIFSTOPPED(status)){
            printf("Job [%d] (%d) stopped by signal %d\n",pid2jid(pid),pid,WSTOPSIG(status));
        getjobpid(jobs,pid)->state=ST;
        }
        else if(WIFEXITED(status)){
            deletejob(jobs,pid);
        }
    }
```

trace 10은 Background 작업이 정상 종료되면 SIGCHLD가 발생하는데, 이 시그널을 받고 Background 작업을 종료 처리한다.
./myspin1 이라는 프로그램을 background 형태로 실행시켜 작업이 종료되기 전에 /bin/kill을 통해서 SIGTERM을 발생시키고 작업이 종료되면 출력문과 함께 joblist에서 제거된다.

Trace 11

```
c201402414@2018-sp:~/shell/shlab-handout$ ./sdriver -V -t 11 -s ./tshref
Running trace11.txt...
Success: The test and reference outputs for trace11.txt matched!
Test output:
#
# trace11.txt - Child sends SIGINT to itself
#
tsh> ./myints
Job [1] (24588) terminated by signal 2
tsh> quit

Reference output:
#
# trace11.txt - Child sends SIGINT to itself
#
tsh> ./myints
Job [1] (24596) terminated by signal 2
tsh> quit
```

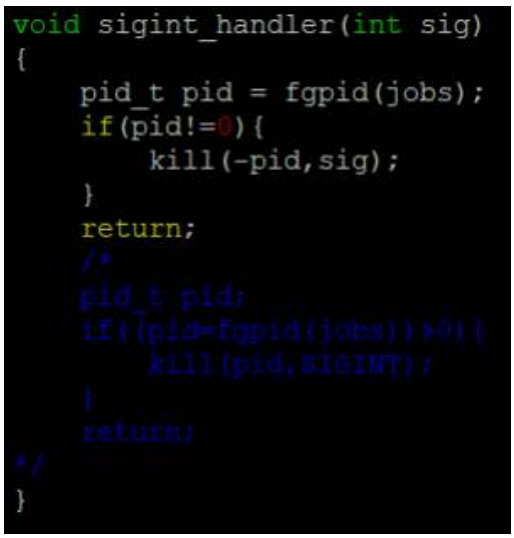
```
c201402414@2018-sp:~/shell/shlab-handout$ ./sdriver -V -t 11 -s ./tsh
Running trace11.txt...
Success: The test and reference outputs for trace11.txt matched!
Test output:
#
# trace11.txt - Child sends SIGINT to itself
#
tsh> ./myints
Job [1] (24623) terminated by signal 2
tsh> quit

Reference output:
#
# trace11.txt - Child sends SIGINT to itself
#
tsh> ./myints
Job [1] (24631) terminated by signal 2
tsh> quit
```

각 trace 별 플로우 차트

-

trace 해결 방법 설명



trace 11은 자식 프로세스 스스로에게 SIGINT 전송되고 처리하는 것을 구현
자신의 pid로 SIGINT를 전당하였을 때 SIGINT에 대한 처리가 되도록 구현하였다.
즉 자식프로세스의 pid로 SIGINT가 전달이 되었을 때 부모의 sigint_handler에 의해서
자식프로세스에 시그널을 보낸다. 위와같이 sigint_handler가 구현되어있어서 trace08을 구현
했을 시 자동으로 성공한다.

Trace 12

```
c201402414@2018-sp:~/shell/shlab-handout$ ./sdriver -V -t 12 -s ./tshref
Running trace12.txt...
Success: The test and reference outputs for trace12.txt matched!
Test output:
#
# trace12.txt - Child sends SIGTSTP to itself
#
tsh> ./mytstps
Job [1] (24946) stopped by signal 20
tsh> jobs
(1) (24946) Stopped ./mytstps

Reference output:
#
# trace12.txt - Child sends SIGTSTP to itself
#
tsh> ./mytstps
Job [1] (24954) stopped by signal 20
tsh> jobs
(1) (24954) Stopped ./mytstps
```

```
c201402414@2018-sp:~/shell/shlab-handout$ ./sdriver -V -t 12 -s ./tsh
Running trace12.txt...
Success: The test and reference outputs for trace12.txt matched!
Test output:
#
# trace12.txt - Child sends SIGTSTP to itself
#
tsh> ./mytstps
Job [1] (24915) stopped by signal 20
tsh> jobs
(1) (24915) Stopped    ./mytstps

Reference output:
#
# trace12.txt - Child sends SIGTSTP to itself
#
tsh> ./mytstps
Job [1] (24923) stopped by signal 20
tsh> jobs
(1) (24923) Stopped ./mytstps
```

각 trace 별 플로우 차트
-
trace 해결 방법 설명
trace12는 자식 프로세스 스스로에게 SIGTSTP를 전송하고 처리하는걸 구현했다.