

Fork3.c

```
1  #include <sys/types.h>
2  #include <unistd.h>
3  #include <sys/wait.h>
4  #include <string.h>
5  #include <stdio.h>
6  int main(void)
7  {
8      int pid, status;
9      char *arg[] = { "./hello", NULL };
10     char ch;
11
12     pid = fork();
13     if (pid > 0) {
14         /* parent process */
15         printf("PARENT: Child pid = %d\n", pid);
16         waitpid(pid, &status, 0);
17         printf("PARENT: Child exited.\n");
18     } else {
19         /* child process */
20         printf("CHILD: Child process image will be replaced by %s\n", arg[0]);
21         execv(arg[0], arg);
22     }
23 }
```

문제 출력 디버깅 콘솔 터미널 포트

```
● (base) 202020827@cslinux:~/os/fork-execv$ gcc fork3.c -o fork3
● (base) 202020827@cslinux:~/os/fork-execv$ ./fork3 &
[1] 1756508
PARENT: Child pid = 1756510
CHILD: Child process image will be replaced by ./hello
● (base) 202020827@cslinux:~/os/fork-execv$ I'm hello. I'm alive!
I'm hello. I'm alive!
ps
  PID TTY          TIME CMD
1755295 pts/112    00:00:00 bash
1756508 pts/112    00:00:00 fork3
1756510 pts/112    00:00:00 hello
1756550 pts/112    00:00:00 ps
(base) 202020827@cslinux:~/os/fork-execv$ I'm hello. I'm alive!
● (base) 202020827@cslinux:~/os/fork-execv$ I'm hello. I'm alive!
I'm hello. I'm alive!
kill 1756510
Command 'kill' not found, did you mean:
  command 'fkill' from snap fkill (v8.0.0)
  command 'tkill' from deb lam-runtime (7.1.4-7)
  command 'xkill' from deb x11-utils (7.7+5build2)
  command 'kill' from deb procps (2:3.3.17-6ubuntu2.1)
  command 'rkill' from deb pslist (1.4.0-3)
  command 'pkill' from deb procps (2:3.3.17-6ubuntu2.1)
  command 'skill' from deb procps (2:3.3.17-6ubuntu2.1)
See 'snap info <snapname>' for additional versions.
● (base) 202020827@cslinux:~/os/fork-execv$ I'm hello. I'm alive!
I'm hello. I'm alive!
kill 1756510
PARENT: Child exited.
[1]+  Done                  ./fork3
● (base) 202020827@cslinux:~/os/fork-execv$
```

설명

- fork() 호출 직후
 1. 부모 프로세스는 `pid > 0` 분기로 들어가 PARENT: Child pid = [자식 PID]를 출력함.
 2. 자식 프로세스는 `pid == 0` 분기로 들어가 CHILD: Child process image will be replaced by ./hello를 출력함.
- 자식 프로세스(execv 실행 후)
 1. 자식 프로세스는 `execv()` 호출로 기존 코드가 대체되어 hello 프로그램을 실행함.
 2. hello 프로그램은 5 초마다 I'm hello. I'm alive!를 출력하며 무한 루프를 돌기 때문에 종료되지 않음.
- 부모 프로세스(waitpid 동작)
 1. 부모 프로세스는 `waitpid()`를 통해 자식이 종료될길 기다림.
 2. 자식(hello 프로그램)이 강제 종료되거나 오류로 인해 종료되면 `waitpid()`가 반환됨.
 3. 이후 부모는 PARENT: Child exited.를 출력하고 main()을 종료함(즉, 부모도 종료).

Fork4.c

```
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int pid, status;
    char *arg[] = { "./hello", NULL };

    while(1) {
        pid = fork();
        if (pid > 0) {
            printf("PARENT: Child pid = %d\n", pid);
            waitpid(pid, &status, 0);
            printf("PARENT: Child exited.\n");
        } else if (pid == 0) {
            printf("CHILD: Child process image will be replaced by %s\n", arg[0]);
            execv(arg[0], arg);
            perror("execv");
            exit(1);
        } else {
            perror("fork");
            exit(1);
        }
    }
    return 0;
}
```

설명

- 부모는 무한 루프 내에서 fork()로 자식을 생성
- 부모는 waitpid()로 자식 종료를 감지하고 메시지를 출력
- 자식은 execv()로 hello 프로그램을 실행하며, hello 는 5 초마다 메시지를 출력
- 자식 종료 시 부모가 새로운 자식을 생성하여 항상 하나의 hello 프로세스를 유지함

```
os > fork-execv > C fork4.c > main(void)
1  #include <unistd.h>
2  #include <sys/wait.h>
3  #include <stdio.h>
4  #include <stdlib.h>
5
6
7  int main(void)
8  {
9      int pid, status;
10     char *arg[] = { "./hello", NULL };
11
12     while(1) {
13         pid = fork();
14         if (pid > 0) {
15             printf("PARENT: Child pid = %d\n", pid);
16             waitpid(pid, &status, 0);
17             printf("PARENT: Child exited.\n");
18         } else if (pid == 0) {
19             printf("CHILD: Child process image will be replaced by %s\n", arg[0]);
20             execv(arg[0], arg);
21             perror("execv");
22             exit(1);
23         } else {
24             perror("fork");
25             exit(1);
26         }
27     }
28 }
```

문제 출력 디버그 콘솔 터미널 포트

```
● (base) 202020827@cslinux:~/os/fork-execv$ gcc fork4.c -o fork4
● (base) 202020827@cslinux:~/os/fork-execv$ ./fork4 &
[1] 1757095
PARENT: Child pid = 1757097
CHILD: Child process image will be replaced by ./hello
I'm hello. I'm alive!
● (base) 202020827@cslinux:~/os/fork-execv$ I'm hello. I'm alive!
I'm hello. I'm alive!
ps
  PID TTY          TIME CMD
1755295 pts/112    00:00:00 bash
1757095 pts/112    00:00:00 fork4
1757097 pts/112    00:00:00 hello
1757159 pts/112    00:00:00 ps
● (base) 202020827@cslinux:~/os/fork-execv$ I'm hello. I'm alive!
I'm hello. I'm alive!
I'm hello. I'm alive!
I'm hello. I'm alive!
I'm hello. I'm alive!
kill 1757097
PARENT: Child exited.
PARENT: Child pid = 1757275
CHILD: Child process image will be replaced by ./hello
I'm hello. I'm alive!
● (base) 202020827@cslinux:~/os/fork-execv$ I'm hello. I'm alive!
I'm hello. I'm alive!
I'm hello. I'm alive!
ps
  PID TTY          TIME CMD
1755295 pts/112    00:00:00 bash
1757095 pts/112    00:00:00 fork4
1757275 pts/112    00:00:00 hello
1757334 pts/112    00:00:00 ps
○ (base) 202020827@cslinux:~/os/fork-execv$ I'm hello. I'm alive!
exec kill -9 -1
□
```

- 프로그램 실행 시, 부모 프로세스는 fork()를 호출하여 자식을 생성하고 "PARENT: Child pid = ..."를 출력함
- 자식 프로세스는 "CHILD: Child process image will be replaced by ./hello"를 출력한 후 execv()로 hello 프로그램으로 대체됨
- hello 프로그램은 5 초 간격으로 "I'm hello. I'm alive!"를 반복 출력함
- 자식 프로세스를 kill 명령으로 종료하면, 부모는 waitpid()가 반환되어 "PARENT: Child exited."를 출력한 후 루프를 돌아 새로운 자식을 생성함
- 새 자식 역시 동일하게 execv()로 hello 를 실행하여 계속해서 하나의 hello 프로세스가 유지됨