운영체제 프로그래밍 과제 – fork & execv	
제목	fork & execv
제출기한	2025/04/02 (수) 15:00
제출내용	1) 수행보고서: 실행 과정, 결과 (화면 캡쳐) 2) 프로그램 소스 코드 (수행보고서에 포함) 유의사항: 프로그램은 반드시 **cslinux**에서 수행해야 함.

1. execv에 대한 설명

설명: 프로그램파일 실행

용법:
execv("실행파일명", 인자리스트);

"실행파일명"	실행 파일 이름 문자열
인자리스트	실행시 주어질 인자들의 배열 (각 인자는 문자열. 첫번째 인자는 실행 파일 이름)

사용 예:

```
char  *argv[] = { "./hello", NULL }; // 인자리스트. 첫번째는 실행파일이름, 마지막은 NULL.
execv("./hello", argv);
```

2. 수행 내용

1) fork3를 실행하고 결과를 설명한다. (fork3.c와 hello.c는 뒤에 첨부되어 있음)

컴파일:

```
gcc  -o  fork3  fork3.c
gcc  -o  hello  hello.c
```

실행:

```
./fork3  &
PARENT: Child pid = 31758
CHILD: Child process image will be replaced by ./hello
I'm hello. I'm alive!
I'm hello. I'm alive!          (5초마다 출력)
. . .
kill  31758                    (자식 프로세스 종료)
PARENT: Child exited.          (자식 프로세스 종료 후 출력)
```

2) fork3.c를 수정하여 **fork4.c를 작성한다**.

fork4의 동작:

fork4의 동작은 fork3와 동일하다. 다만, 자식 프로세스가 종료되는 사건이 발생한 경우, 자식 프로세스를 (fork를 통해) 자동으로 다시 생성하고 (execv를 통해) hello를 실행시킨다. 즉, (hello를 실행하는) 자식 프로세스 1개가 항상 실행 중에 있도록 한다.

실행 예:

```
./fork4  &
PARENT: Child pid = 31758
CHILD: Child process image will be replaced by ./hello
I'm hello. I'm alive!
I'm hello. I'm alive!          (5초마다 출력)
. . .
kill  31758                    (자식 프로세스 종료)
PARENT: Child exited.          (자식 프로세스 종료 후 출력)
PARENT: Child pid = 31770      (새로운 자식 프로세스 생성 후 출력)
CHILD: Child process image will be replaced by ./hello
I'm hello. I'm alive!
I'm hello. I'm alive!          (5초마다 출력)
. . .
```

3. 제출 사항

수행 보고서: 문서 파일로 4/2(수) 15:00까지 제출

- fork3 실행 결과에 대한 설명
  (각 출력이 나오는 시점, 자식프로세스 종료 후 동작 등)
- fork3 실행 화면 캡쳐
- fork4.c 소스코드 및 설명
- fork4 실행 결과에 대한 설명
  (각 출력이 나오는 시점, 자식프로세스 종료 후 동작 등)
- fork4 실행 화면 캡쳐

4. 주의

1) 과제는 반드시 본인 힘으로 완성해야 함
2) 프로그램 버그로 fork 횟수가 많아져 프로세스가 개수가 너무 많이 증식하는 경우, 이를 아래와 같이 제어하기 바랍니다.

```
exec  kill  -9  -1
```

(내 계정의 모든 프로세스를 종료. 관리자 계정에서는 사용하지 말 것.)

[fork3.c]

```
/*
 *  Fork a child process and execute a new program code.
 *
 */

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

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

    pid = fork();
    if (pid > 0) {
        /* parent process */
        printf("PARENT: Child pid = %d₩n", pid);
        waitpid(pid, &status, 0);
        printf("PARENT: Child exited.₩n");
    } else {
        /* child process */
        printf("CHILD: Child process image will be replaced by %s₩n", arg[0]);
        execv(arg[0], arg);
    }
}
```

[hello.c]

```
/*
 *  A file executed by child process
 *
 */
#include <unistd.h>
#include <stdio.h>
main()
{
    do {
        printf("I'm hello. I'm alive!₩n");
        sleep(5);
    } while (1);
}
```